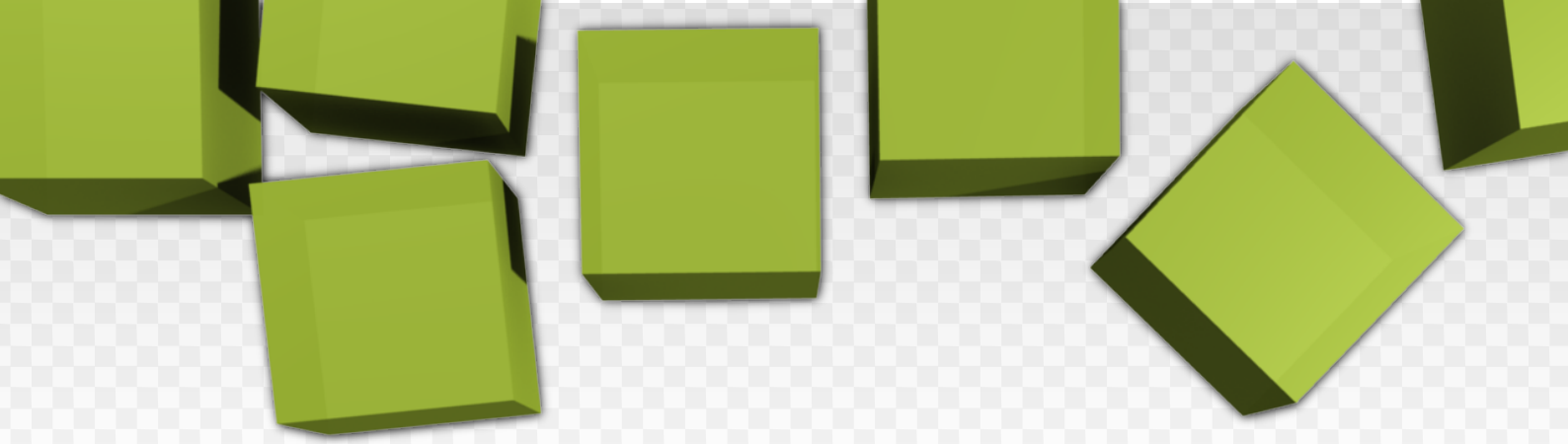


3

IMPLEMENTATION OF FREE SOFTWARE SYSTEMS

AUTHOR:
A. Albós Raya
M. D'Elia Branco
M. León Martínez
A. Novo López
A. Otero García
Ó. Sánchez Jiménez

COORDINATOR:
D. Megías Jiménez

FREE TECHNOLOGY ACADEMY

Preface

Software has become a strategic societal resource in the last few decades. The emergence of Free Software, which has entered in major sectors of the ICT market, is drastically changing the economics of software development and usage. Free Software – sometimes also referred to as "Open Source" or "Libre Software" – can be used, studied, copied, modified and distributed freely. It offers the freedom to learn and to teach without engaging in dependencies on any single technology provider. These freedoms are considered a fundamental precondition for sustainable development and an inclusive information society.

Although there is a growing interest in free technologies (Free Software and Open Standards), still a limited number of people have sufficient knowledge and expertise in these fields. The FTA attempts to respond to this demand.

Introduction to the FTA
The Free Technology Academy (FTA) is a joint initiative from several educational institutes in various countries. It aims to contribute to a society that permits all users to study, participate and build upon existing knowledge without restrictions.

What does the FTA offer?
The Academy offers an online master level programme with course modules about Free Technologies. Learners can choose to enrol in an individual course or register for the whole programme. Tuition takes place online in the FTA virtual campus and is performed by teaching staff from the partner universities. Credits obtained in the FTA programme are recognised by these universities.

Who is behind the FTA?
The FTA was initiated in 2008 supported by the Life Long Learning Programme (LLP) of the European Commission, under the coordination of the Free Knowledge Institute and in partnership with three european universities: Open Universiteit Nederland (The Netherlands), Universitat Oberta de Catalunya (Spain) and University of Agder (Norway).

For who is the FTA?
The Free Technology Academy is specially oriented to IT professionals, educators, students and decision makers.

What about the licensing?
All learning materials used in and developed by the FTA are Open Educational Resources, published under copyleft free licenses that allow them to be freely used, modified and redistributed. Similarly, the software used in the FTA virtual campus is Free Software and is built upon an Open Standards framework.

Evolution of this book

The FTA has reused existing course materials from the Universitat Oberta de Catalunya and that had been developed together with LibreSoft staff from the Universidad Rey Juan Carlos. In 2008 this book was translated into English with the help of the SELF (Science, Education and Learning in Freedom) Project, supported by the European Commission's Sixth Framework Programme. In 2009, this material has been improved by the Free Technology Academy. Additionally the FTA has developed a study guide and learning activities which are available for learners enrolled in the FTA Campus.

Participation

Users of FTA learning materials are encouraged to provide feedback and make suggestions for improvement. A specific space for this feedback is set up on the FTA website. These inputs will be taken into account for next versions. Moreover, the FTA welcomes anyone to use and distribute this material as well as to make new versions and translations.

See for specific and updated information about the book, including translations and other formats: *http://ftacademy.org/materials/fsm/*. For more information and enrolment in the FTA online course programme, please visit the Academy's website: *http://ftacademy.org/*.

I sincerely hope this course book helps you in your personal learning process and helps you to help others in theirs. I look forward to see you in the free knowledge and free technology movements!

Happy learning!

Wouter Tebbens

President of the Free Knowledge Institute

Director of the Free technology Academy

The authors would like to thank the Foundation for the Universitat Oberta de Catalunya for financing the first edition of this work, and a large share of the improvements leading to the the second edition, as part of the Master Programme in Free Software offered by the University in question, where it is used as material for one of the subjects.

The translation of this work into English has been made possible with the support from the SELF Project, the SELF Platform, the European Comission's programme on Information Society Technologies and the Universitat Oberta de Catalunya. We would like to thank the translation of the materials into English carried out by lexia:park.

The current version of these materials in English has been extended with the funding of the Free Technology Academy (FTA) project. The FTA project has been funded with support from the European Commission (reference no. 142706-LLP-1-2008-1-NL-ERASMUS-EVC of the Lifelong Learning Programme). This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Acknowledgements

The authors would like to thank the Foundation for the Universitat Oberta de Catalunya (http://www.uoc.edu) for the funding of this project, which forms part of the International Master's Degree in Free Software offered by the University.

They would also like to thank Jordi Mas for his help in coordinating the first edition of this material.

Annex

GNU Free Documentation License

GNU Free Documentation License
Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any

member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent.

An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of

transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front

cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition.

Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material.

If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number.

Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit.

When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form.

Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their

copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME.

Free software implementation, projects and companies

Amadeu Albós Raya
Óscar David Sánchez Jiménez

PID_00148382

Universitat Oberta de Catalunya
www.uoc.edu

Index

Introduction

This module looks in detail at the methodology used to implement free software systems in organisations and generic scenarios, establishing the main features that will guide the project and its development.

Technology is now becoming a strategic factor in both public and private organisations, whether small, medium or large. The integration of technology in all of the organisation's functional and operating processes means that its state of operation is closely related to the production of the organisation. As a result, we need to systematically monitor the efficiency and effectiveness of the system in order to control performance in line with the evolution of the strategic needs of the organisation.

The implementation of systems cannot be left at random or to the convenience of factors that are irrelevant to the organisation because the results can be unexpected and have consequences of varying magnitudes for the organisation. Hence, we can see the need for a rigorous implementation of systems with the methodology used in scientific and technological circles, by which we establish a framework of support with guarantees for management of the complexity, development of the project and management of the risks inherent both to the technology and the system implementation process.

The main aim of this module is to provide a broad and detailed vision of the main processes related to the implementation of free software systems, both in a generic or abstract context and from a perspective focusing on the characteristics of the functional and operational management of the project to implement the system, or considering the free software business as a valid and viable option for a profitable company.

All of the above must be subject to a methodical and rigorous management allowing, on the one hand, satisfactory management of the complexity of solving the specific problems of the systems implementation project and, on the other, keeping under control all of the potential risks that could cause the project to fail in one way or another, whether prematurely or while the system is already in operation, both important consequences for the organisation and its users.

The contents of this module are divided into three units that gradually introduce the key theoretical concepts of the process of implementing a system in an organisation. Its sections present the implementation of systems from the perspective of free software, analysing the main features that will produce a successful project both from a project management perspective and

from that of the implementation of free software as a profitable business. A brief summary then follows of the main features of each of the units in this module.

The first section, "Introduction to systems implementation", gives a general presentation of systems implementation and the features of free software. We define the term "systems implementation" as an action resulting from the strategy of the organisation, detailing the different reasons leading to implementation, the main stages of the process and a classification of the main typologies based on project context and aim. We present the specific features of free software, their effects on systems implementation and the functional management of the project in terms of scope, time, integration, cost, quality, human resources, communication and risks.

The second section, "Free software projects", presents the free software implementation project in detail from the point of view of its methodology. We define the life cycle of the project and its main characteristics, repercussions and relationship with the aims. The stages of the implementation project are described in detail: study of the current situation, study of implementation issues, analysis of free software solutions, formalising of the proposal, development, implementation and migration, and lastly, training, communication and support for users. Each of the stages is defined and described in detail in relation to its specific and global objectives.

The third section, "Free software companies", provides a detailed description of free software companies as a valid and viable alternative to sales of copies of software. We present the main business models for free software, based primarily on the production and sale of extra services. Lastly, we take a detailed look at the features of the business plan of free software companies, focusing on aspects such as definition, scope, organisation, human resources and materials, production, product evolution, quality control and monitoring, guarantees and user support, economics, financing and the project feasibility study.

We hope that the conceptual, methodological and practical formalising of the main aspects of the project to implement free software systems will enable you to understand the need for and importance of the process and the actions that must be taken to guarantee the strategic aims of the project.

Objectives

The aims of this module are:

1. to be aware of the basic concepts of the implementation of free software systems;

2. to be able to identify the different types of project to implement free software systems;

3. to be familiar with the project management areas and their specific features in application to free software;

4. to know the main elements of free software projects;

5. to know the life cycle phases of free software projects;

6. to learn how to draft a free software project proposal;

7. to obtain an in-depth knowledge of the features of free software migration projects;

8. to learn how to plan and execute free software migration projects;

9. to know the business models used by free software companies;

10. to become familiar with the main elements of a business plan;

11. to learn how to draw up a business plan based on free software; and

12. to know the main characteristics of free software production and its specific features.

1. Introduction to the implementation of free software systems

This first module of this subject, *Implementation of free software systems*, provides the basis for discovering the main concepts of the implementation of systems in general, and its application to free software in particular.

The module begins with an overview of the basic concepts of systems implementation projects, characterising the different phases of the process from a methodological and functional point of view and analysing the close relationship between the strategic objectives of the organisation and the implementation project.

We then provide a classification of the most common types of implementation project, indicating their basic characteristics and the differences between them, and carefully analysing the implications of this for the objectives and performance of the project.

After summarising the basic concepts of systems implementation and the different types of project it includes, we look at the specific features of the implementation of free software systems. We analyse the main factors influencing the project and the pros and cons of implementation with free software.

Lastly, we look at the basic concepts of project management and expand on the classical models in order to adapt them to the specific features of free software implementation.

1.1. Basic concepts

This first section provides an overview of the basic concepts of systems implementation, from conceptual definition to its main phases, including methodological, strategic and organisational aspects.

1.1.1. Definition

Systems implementation has always been closely associated with the evolution and popular diffusion of technology. In fact, if we consider this concept from a global perspective, any innovation – technological or otherwise – that we wish to extend beyond the borders of the context of its creators needs to undergo an implementation process.

When implementing technology, we need to pay attention to certain basic aspects, such as the impact on the organisation and direct users, but also on indirect users and clients, for example. Furthermore, technology – in the global and generic sense – is currently seen as a decisive factor in the competitive evolution of organisations.

Systems implementation is the process by which one or more technological innovations are introduced into an organisation, as the result of an action deriving from its strategic plan.

In line with this definition, the implementation of technological systems is the result of an organisation's strategic desire to reach new milestones, whose aims can be very wide-ranging, depending on the context of the organisation.

We can illustrate this concept with two examples from different contexts:

- Companies, as profit-making organisations, can take action concerning the technology they use in an attempt to increase their competitiveness and obtain greater market shares, thus being able to offer more innovative products or ones tailored to new demands.

- Government agencies, as non-profit-making organisations, can take action concerning the technology accessible by the region they govern in an attempt to provide competitive tools that will reduce the digital divide and develop the sector's economy.

1.1.2. The organisation's strategic plan

In the previous section we mentioned the organisation's strategic plan and its important role in systems implementation.

An organisation's strategic plan is a series of proposals setting down the future aims or directions of the organisation. It usually covers a five year period and is developed in the diverse divisions or functional departments of the organisation.

The main aim of the plan is to minimise the risks and maximise the results of implementation by marking out real, affordable and measurable directions generated by the relationship between the organisation and the area in which operates.

Very often, SWOT[1] analysis is used to diagnose the organisation's current situation. This tool summarises in a single table the main factors influencing the structural operation of the organisation in its context at a given time.

(1)SWOT is an acronym for *strengths*, *weaknesses*, *opportunities*, *threats*).

Considering that the system implemented has to meet the current and future needs of the organisation, we can understand the importance of the link between the organisation's strategic plan and the implementation project.

The organisation's strategic plan evolves over time and adapts dynamically to changes in the context in which it is carried out. Additionally, the organisation's system must evolve with strategic changes and must conclude some of the actions begun with a new systems implementation project.

1.1.3. Origin of systems implementation

In the preceding paragraphs, we saw the link between the implementation project and the organisation's strategic plan. Organisations do not introduce implementation projects without first determining that they are necessary for their specific strategies.

Thus, the implementation of a system requires having detected deficiencies in the organisation's current system, although they can also be implemented in new organisations or those without prior technological positioning.

Broadly speaking, there are four generic origins leading to the implementation of a new system:

- **Detection of problems**: there may be diverse cases in which the current system operates inefficiently, which compromises the everyday tasks of its users and the reliability of the system.
  In these cases, the strategic plan is chiefly affected by loss of performance and efficiency in the organisation.
  An example of this type of situation could be programming errors that produce inaccurate calculations, access errors or system locking.

- **System evolution**: this covers situations in which the current system is functionally obsolete, which compromises the organisation's operation due to the lack of features to solve the increasing number of issues in the organisation.
  In these cases, the strategic plan is affected by the organisation's loss of effectiveness. An example of this type of situation might be the need to increase features following a change in legislation.

- **System enhancement**: in these situations, the system in place is structurally obsolete, which compromises the organisation's operation due to the poor performance of the platform of the current system.
  In these cases, the strategic plan is affected by the loss of performance and efficiency of the organisation. An example of this situation might be the lack of integration of new operating systems or different types of hardware.

- **New strategic action**: this includes possible updates, changes or innovations in the organisation's strategic plan not covered by the current system.
  In these cases, the strategic plan is affected by the organisation's loss of effectiveness. An example of this type of situation might be an increase in the services offered or expansion of the target market.

The above list of possible origins is not all-inclusive but it does indicate the most logical reasons for systems implementation. In addition, the different cases are not mutually exclusive and their coincidence can be strongly motivated by the evolution of the organisation and its system.

1.1.4. Resources of a systems implementation project

Normally, once the need to adapt the current system to the organisation's strategic plan has been noted, resources are allocated to the new systems implementation project. Initially, these resources usually take the form of one or more individuals with free time to spend on the project, with the financial repercussions that this will have on the everyday workings of the organisation (insourcing[2]).

(2)The term insourcing refers to the internal delegation or production of a process.

Some organisations prefer to leave this task to professionals outside the company (subcontracting or outsourcing[3]) for reasons of functional objectivity, production capacity or time availability. In these instances, it is not the case that the organisation spends no time at all on the project; rather, this time is reduced by the degree to which outsourcing increases, since both the organisation and the external professionals need one another to bring the project to successful completion.

(3)The term *outsourcing* refers to the external delegation or production of a process.

One of the key points of an implementation project, regardless of its final form of execution, is the creation of a supervisory or monitoring committee for the project (executive committee in some organisations). This committee is charged with the methodological execution of the project and its adequate, gradual and sustained progress over time.

The supervisory committee is normally made up of individuals from the various divisions affected by implementation, mainly management staff and heads of department. If the organisation contracts external professionals to

manage the implementation, they will also form part of the committee. Although most of the committee members only spend part of their time on it, there is usually at least one member that works on the project full-time to ensure rigorous monitoring.

The importance of resources in the implementation of a system is twofold:

- Firstly, human resources are allocated based on the quantity and quality in the analysis and design of the system implementation.

- And secondly, material resources are allocated according to the quality and quantity of the system to be implemented.

In all events, the resources allocated to a systems implementation project will have a direct effect on the finances of the organisation.

1.1.5. The main stages of a systems implementation project

As we explained above, a systems implementation project is a methodological process designed to adapt the system to the strategic plan. This process must be performed with due care and attention in order to guarantee the success of the project.

From a generic point of view, we can break the systems implementation process down into four main phases: analysis of the current system, design of the new system, development, and implementation of the system.

Analysis of the current system

All implementation projects begin with a study and analysis of the current status of the organisation in the terms indicated by the strategic action. There are two possible initial situations:

- If the organisation has a system in place, its characteristics are assessed by collecting information on the elements affecting the strategic action and their structure. The aim of this is to create an abstract framework for adapting the system to the new strategy.

- If the organisation does not have a system in place (or it is a new organisation), we need to evaluate the features of the area of action that will be affected by the strategic scope of the organisation's operation. We will need to create an abstract framework of aims that the implementation project must meet.

In either case, this stage defines and determines the problems that the implementation project will need to overcome, given that we will select the different aims of the strategic action. We evaluate aspects such as system history, structure and operation or the evolution of issues and workload over time. Many of these results are presented in the form of charts or diagrams.

The stage ends with a presentation of the conclusions of the study and an analysis of the current status, which evaluate the extent to which the system can stand up to new strategic challenges.

Design of the new system

Once we have assessed the initial situation of the organisation, defined the main points of the strategic action and received confirmation of monitoring from the project's executive committee, we can begin to design the new system.

This stage begins with a thorough analysis of the issues that the new implementation will need to resolve, based on the strategic actions we have defined. We will come up with different solutions or alternatives for these problems that we will need to analyse individually to gauge their suitability and determine their costs, advantages and disadvantages, both tangible and intangible. Depending on the type of project and strategic action, this evaluation will cover a five-year period.

We must be objective and use a methodology when choosing our solution in order to maximise the advantages and minimise the disadvantages both of implementation of the solution and its everyday operation. Some decision-making criteria may concern the scope of the action, performance, the necessary equipment, the requirements covered, provider support, availability of equipment and support, and the maintenance required over time.

Development of the new system

After receiving confirmation of the project monitoring by the supervisory committee, we can begin to develop the new system.

The development of the system or adaptation of the proposed solutions adopts a life cycle most suited to the purpose of the project, some of which we have studied in other subjects. At the end of this stage, we obtain a system ready to be implemented in the organisation that will resolve the strategic issues observed in the previous stage.

See also

The subject on *Software engineering in free software environments* contains more information on the life cycle of software development.

Implementation of the system

Once the new system is ready, we can begin the implementation phase, which involves setting up the system in the organisation.

This phase sees the completion of the adaptation and integration of the new system into the real environment and covers user training, pilot tests and system integration with end user testing, and the conversion and final release of the new system.

If the organisation already has a system in place, we must also take into account the migration from the old system to the new one. Migration involves transferring the current status of the system in place to the new system. Data transfer is usually the most important task of migration, since this activity cannot affect the day-to-day running of the organisation.

This stage ends with the final implementation of the new system, data migration and user training. In other words, we will have fully introduced the elements needed for performance of the strategic action we defined at the start.

Although a considerable proportion of the implementation project is completed with the implementation of the system, the maintenance and continuous evaluation of the system's feasibility remain active as in any other project, particularly if the new implementation is regarded as a strategic factor in the competitiveness of the organisation.

1.1.6. Feasibility and evaluation of the project

The previous section outlined the main phases of a systems implementation project but we also need to consider the importance of at least two control points for the project.

The first is the feasibility and continuation of the project, which is assessed on the basis of two milestones:

- The first milestone occurs after the current status analysis stage, in which we analyse and discuss the current system status in terms of the strategic action.

- The second milestone takes place after the design stage, which involves an analysis and discussion of the proposed solutions.

For instance, the convenience of continuing the project could be questioned given the financial implications of its development.

The second control point is the project evaluation, which takes place once the system has been fully implemented in the organisation. At this milestone, now that implementation is complete, we analyse the true repercussions of the new system by evaluating the measurable impact factors described in the strategic action that led to the implementation process. The aim is to evaluate the new system's concordance with the strategy of the organisation. These measurable indicators are observed from a longitudinal perspective.

Longitudinal perspective

Observing measurable indicators from a longitudinal perspective means that the observation is not unique in time but is in fact repeated several times over a previously defined period.

For example, we can evaluate the impact of the new system on the organisation's production capacity or on its ability to attract new clients.

1.1.7. Project methodology

As in all strategic projects, systems implementation must be carried out in a methodological, structured and orderly way. The importance of the project outcome for the development of the organisation dictates the precision with which it needs to be carried out.

To illustrate this concept, we can contrast it with the methodology applied to life cycle in software development and the importance of the conceptual analysis and solution design phases. Here we can see the problems caused by not detecting failures in the analysis and design stages. Solving a design problem detected at the development stage is generally considered to have a possible cost ratio of one to ten.

Systems implementation projects also require maximum precision in the analysis and design phases because of the strategic implications of the new implementation on the organisation's evolution.

The stages described in the previous sections have a sequential behaviour, inherent to their aims, but we can still specialise the methodology for execution of each stage in order to achieve our aims more efficiently. One example would be the distribution below:

- The current system analysis phase may use an iterative method, with feedback obtained from the study results.

- The design and development phases may use an XP[4] method if we are developing software, or a classical schema if we are implementing an infrastructure.

- The implementation phase can use an iterative process if it affects a number of units, with the possibility of maintaining more than one line of implementation for each time unit.

(4)XP stands for eXtreme Programming, an agile software development methodology.

In all events, we will need to adapt the methodology of each stage to the convenience of the project and the organisation, always with the aim of maximising efficiency and minimising any negative impact.

1.2. Types of project

This section contains a general classification of the different types of implementation project and describes the diverse features and main implications for project development.

It offers two classifications, one based on project scope, that is, the repercussions and scope of the implementation, and a second that considers the aim of the project, that is, the contents of the implementation.

This classification should not be regarded as a rigid structure, since the true diversity of the projects allows a combination of different typologies.

1.2.1. Classification by scope

We can define three broad scopes of action for implementation projects:

1) Internal

Projects whose scope is internal are designed to implement a local system in an organisation that will primarily be used internally.

Examples of projects with an internal scope are those implementing local network services (directory, authentication or data sharing services), those implementing tools in local groups (internal mail or groupware) or those implementing internal management tools (ERP systems).

The strategic aim of this type of project is to introduce functional internal improvements to the efficiency and efficacy of work through technological renovation. These are normally used in scenarios in which the current system reveals functional deficiencies or is unreliable.

The main implications for project development are as follows.

- **Evaluation of the current system**: This stage basically evaluates the system from a functional perspective, pinpointing and evaluating the aspects of the system that could cause problems, always taking the strategic action outlined by the organisation as a reference.

- **Design of the new system**: The issues analysis carried out at the design stage must allow us to compare two alternatives for implementation:

on the one hand, evolution of the current system, and, on the other, a complete overhaul of this system.
While evolution seeks to partially update the current system in order to extend its useful life, a complete overhaul requires changing various elements of the system (hardware and/or software).

- **Development of the new system**: There are no major differences in the development stage. In all events, the local scope of the project can help us to define the latter and provide a strategic resolution.

- **Implementation**: The project's internal scope can help with progressive implementation (of time and/or services) which, along with staff training, must contribute to the creation of a mood of acceptance while guaranteeing the day-to-day operation of the organisation.

2) External

The aim of external projects is to implement a system essentially for public use in the organisation, connecting external agents to the organisation. The system can be placed locally or remotely.

Examples of projects with an external scope are those that implement corporate intranets or extranets (tailored access to workers, clients or suppliers) or implementation of electronic government services (e-government or e-voting).

The strategic aim of this type of project is to introduce functional improvements to the organisation in its external dealings, enhancing the efficiency and efficacy of its communication through technological renovation. They are normally used in scenarios in which the management of digital relations with third parties is complex or inefficient and in cases where we are looking to improve the corporate image and target market.

The main implications for project development are as follows.

- **Evaluation of the current system**: This stage basically evaluates the system from a relational, communicative and interactive perspective. Data collection cannot be limited to the internal part of the organisation, so contact with external agents will be required. Regardless of whether or not there is a system in place, the aim of the strategic action will be to highlight the shortcomings and weaknesses of the current relational implementation.

- **Design of the new system**: The design of the new system basically needs to cover two types of aim: the efficiency and efficacy of user functionality,

and considerations relating to the corporate image that the organisation wishes to convey to the target market.

- **Development of the new system**: There are no major differences in the development stage. However, it should be noted that security management is particularly important with this type of project.

- **Implementation**: The full implementation of these systems can be carried out in two phases: we can first set up the basic system and then we can progressively update the services and contents over time, taking into account the opinions of external users on the changes that have taken place (feedback).

3) Productive

The aim of projects with a productive scope is to implement a system in a different environment to that of the organisation managing and/or developing the project (outsourcing).

Examples of projects with a productive scope include core software implementation (operating systems), specialist software implementation (office automation tools, accounting, invoicing, etc.) and implementation of outsourced services (subcontracting of website services).

The strategic aim of this type of project is to meet the demands or needs for technology services of other organisations or external groups from diverse fields. It is normally used in scenarios associated with opportunities for a strategic and technological change in the target market.

The main implications for project development are as follows.

- **Evaluation of the current system**: This stage basically evaluates the system from two different angles. Firstly, specialised projects for an organisation with specific strategic needs to be met.
  And secondly, direct implementation projects (operating systems or office automation tools), designed to provide generic solutions for a large market segment. In both cases, the importance of studying the status of the current situation is clear.

- **Design of the new system**: There are no significant differences in the design stage. Specialised projects are carried out in accordance with the internal or external project considerations in the client organisation while direct implementation projects need to meet everyday operating needs

and/or offer new features. The technology used in the project is important for the corporate image of the organisation.

- **Development of the new system**: There are no major differences in the development stage. It is important for the development to meet the aims of the client organisation efficiently and effectively, and that this is done within the fixed time limit.

- **Implementation**: Specialised projects are implemented based on typology (internal or external project), with an emphasis on user training and communication. Direct implementation projects are normally implemented as a self-installing product package (on CD-ROM or DVD-ROM, or as a download from the Internet). Some services can be considered extras (training, migration, etc.) and supplied on request by the organisation or by third parties.

1.2.2. Classification by requirement aim

Three main groups of aim can be defined for implementation projects:

1) Software

The aim of software development projects is to implement applications to meet certain demands. This type can also include projects for the adaptation, reengineering or integration of software or tools, such as the adaptation of operating systems or the integration of specialised software packages.

The strategic aim of this type of project is to provide a technological response to a specific functional problem. It is normally used in scenarios where systems are implemented to automate tasks, provide efficient technological support to users or to evolve or replace obsolete software.

The main implications for project development are as follows.

- **Evaluation of the current system**: This stage basically evaluates the system from a functional perspective, evaluating overall efficiency and efficacy in terms of the strategic action that launched the project.
  In projects involving the creation of new or generic software, the evaluation focuses on the study and analysis of the current status of the target market and its main directions.

- **Design of the new system**: There are no significant differences at the design stage in the life cycle of software production that we have studied in other subjects. The use of methodology is important during the analysis of needs and system design phases to minimise the errors detected at later stages.

See also

For more information on the life cycle of software production, see the subject on *Software engineering in free software environments*.

- **Development of the new system**: There are no major differences in the development stage regarding the usual life cycle of this type of project. Aspects of importance include the mechanisms guaranteeing the quality of the code produced and the evolution of the code in versions and revisions.

- **Implementation**: We can distinguish between two possibilities at the system implementation stage. On the one hand, the development of software that needs to be installed in an organisation, which will follow the usual process, taking into account the possible need for migration if there is a system already in place. And on the other, common or generic software, which is usually implemented as a self-installing product package on a CD-ROM or DVD-ROM, or as a download from the Internet. Some services can be considered extras (training, migration, etc.) and supplied on request by the organisation or by third parties.

2) Infrastructure

The aim of infrastructure implementation projects is to implement architectural or structural systems in a given environment. These projects are usually used to implement functional equipment that provides a basic service for the organisation.

Functional equipment providing basic services

This applies to the installation and configuration of servers or clients, such as operating systems, office automation software, basic network services (DNS, DHCP, etc.) and advanced network services (mail, groupware, etc.).

The strategic aim of this type of project is to provide a functional technological basis to meet the requirements of the organisation. It is normally used in scenarios of new creations or technological renovation due to the system becoming obsolete. We need to remember that the organisation's infrastructure is its functional basic architecture, on which the rest of the technological elements will be organised.

The main implications for project development are as follows:

- **Evaluation of the current system**: This stage basically evaluates the system from a functional perspective, taking into account aspects such as efficiency and efficacy as well as reliability and adaptation to the new standards of technological evolution.

Where there is no previous system in place, we will basically need to take into account the needs (quality and quantity) of the organisation, together with current directions and standards.

- **Design of the new system**: The design stage deals mainly with the research and study of the diverse solutions available on the market, although we cannot rule out the creation or evolution of a solution if the needs of the organisation so require. By comparing these options, we should come up with the product that best fits the strategic action, taking into account evaluation criteria such as the scope, efficiency and efficacy of the solution, equipment and support requirements, availability and product maintenance.

- **Development of the new system**: The system development stage involves the preparation of procedures to implement the infrastructure, bearing in mind that it is sometimes necessary to tailor the product or adapt configuration files.

- **Implementation**: If the new system replaces an earlier one, we will need to migrate from one system to the other. Nonetheless, we can split implementation into two phases: firstly, the installation and configuration of the new system features and secondly, start up and status restore.
  User training and communication are essential for this type of project to ensure acceptance of the new features and for an assessment of its operation after start-up.

3) Systems migration

The aim of systems migration projects is to transfer the status of the current technological architecture to a different one. These projects are implemented when one or more main elements of the system are updated.

The strategic aim of this type of project is to minimise the impact of technological changes on the operation of the organisation. It is normally used in scenarios where new software systems or infrastructures are implemented, but they can also be developed independently to replace the physical technological platform in order to improve performance, reliability or in the event of obsolescence.

The main implications for project development are as follows:

- **Evaluation of the current system**: This stage evaluates the system primarily from the point of view of preserving the configuration and

stored data. It is important to adopt a methodological approach in the research and evaluation of the various elements that need to be migrated to the new system.

- **Migration design**: The design stage normally involves studying and analysing the methods and procedures that we will need to implement to transfer the status between the two systems. We can consider aspects such as the preparation of backups or the design of procedures to export or convert the data in the current system.

- **Migration development**: Migration development requires the performance of all tasks to preserve the data and configuration, and to export and convert the current system. With this task, we must take into account the importance of the temporary location, to ensure that data is transferred fully without interrupting the day-to-day operation of the organisation. We also need to consider the security guarantees of the backup medium.

- **Implementation**: Implementation covers start-up of the new system and the status restore of the organisation's previous system. The timing of this process must be planned in conjunction with the migration development stage in order to minimise the impact of the change on the day-to-day operation of the organisation, although the progressive restoring of some minor elements may be possible. Communication and collaboration with users is particularly important in this type of project if we are to meet the aims of migration successfully.

1.3. Free software systems

This section introduces the specific features of the implementation of free software systems. We will analyse the main factors affecting the project and the myths, advantages and disadvantages of using this type of software.

When we talk about free software, we generally refer to its advantages, which are fairly well-known. For instance, free software is secure and good quality, is distributed freely, uses open standards and is based on the culture of collaboration and promotes the latter. What is more, it can be used anywhere, enhances technological capabilities, helps to reduce overheads and operating expenses in IT systems, reduces dependence on providers and fosters the development of local companies.

However, the companies that implement free software systems come up against a series of problems that is hindering the sector's development (Sáez et al).

Generally, to encourage the development of a technology, we need this technology to be commercially viable – i.e. there has to be a supply and demand – and economically feasible, in that the companies from that sector must generate profits through their implementation of the technology.

With **demand**, that is, the companies that could adopt the free software, the main obstacles are piracy, fear of change and misgivings. Companies confuse free software with freeware, and some companies rule out its implementation either because free software of similar standards is not available or because they do not believe that there are companies behind this software guaranteeing its support and maintenance.

With **supply**, that is, the companies that provide the applications and free software services, the main obstacles are fear of change and the lack of cooperation.

On the whole, IT companies are used to developing tailored software solutions without giving their clients the sources, and setting up a model of licence payments for their use. And yet, with the appearance of free software, some companies are seeing that a business model based on payment for services rather than licences can be sustainable, leading them to release part of the code that was thus far kept private. Therefore, one possibility that could be used initially by many companies would be to offer two solutions to clients: ownership (with a licence fee) and free (normally without a licence fee). This is just one of many possible business models for free software.

IT companies are also used to developing software on an individual basis but a better option would be to develop the cooperative model. This would involve drawing on work done by others, collaborating, and transferring licence savings to the end client.

Free software projects may be approached as normal projects from the point of view of software engineering and project management. However, a closer look will reveal certain differences and specific features that are often only dealt with correctly with experience and whose omission can affect and even bring about the failure of some projects.

In the light of this, the use of a comprehensive methodology for the implementation of free software systems is essential, particularly because:

- It gives **clients more confidence** in the quality of the products and processes, whether we are developing a new programme or application, migrating an existing system or starting up a new one.

- It allows providers to systematise the procedure for **the implementation of free software systems** and become familiar with their features, which results in improved efficiency and allays fears of change.

1.4. Management of free software projects

Project management is traditionally divided into nine areas of knowledge:

- Scope
- Time
- Integration
- Cost
- Quality
- Human Resources
- Communication
- Risks
- Supplies

This section will look at each of these in turn and study their specific features when applied to projects implementing free software systems.

1.4.1. Scope management

Scope management involves defining the aims of the project so that we can check to see that they are being met and, if needed, change them. In other words, scope management ensures that the project carries out all necessary work – and only the necessary work – so that the initial aims can be met.

Definition of the project scope

To define the scope of a project, the project manager must establish the project work breakdown structure (WBS), which divides the project into work packages, usually represented in the form of a logical tree. A work package is the smallest unit into which a project can be divided in order to make it independently manageable.

We therefore need to identify everything that needs to be done in the project through its WBS, briefly describing its work packages and the deliverables that each one needs or facilitates.

Changes to project scope

It may sometimes be necessary to modify the scope and aims of the project during its execution. This can be due to the following:

- Shortcomings in the original project plan, especially incorrect definition of the scope.
- Changes to the needs and requirements of the client established in the initial project plan.
- Changes to the context of the project and hence, to the hypotheses considered when the initial project plan was drawn up.

These contingencies may have a significant impact on project execution, modifying and even preventing it from achieving its aims. Thus, it is essential to control changes and to take into account risk management.

Scope management in free software projects

The features of scope management in free software projects are the same as those of any other software project. It is essential to obtain and conduct a detailed analysis of client requirements – and of the current situation of the system if dealing with a migration project.

The definition of the scope of free software projects will generally depend on the motivation for the project: cutting costs, system improvement, independence of distributors or regularising the software licence situation.

Lastly, it is important to note that clients may not always be aware of the consequences of changes to the project once it is underway.

1.4.2. Time management

The purpose of time management is to ensure that the project is carried out within the set deadlines. This means that we will need to define the sequence of activities to be performed, along with their duration and coordination.

Good time planning is an essential part of project management because it establishes the model by which the project will be carried out. Moreover, it allows us to ensure that the aims are being met, it lays the foundations for integrating time, costs and resources, and it sets down a common framework for the various individuals and partners taking part in it.

Project network

The WBS we saw in the previous section is used to identify the activities needed to conduct the project, taking into account the fact that an activity is the smallest part of work into which a project can be divided for planning purposes.

Following on from this, we need to identify the order in which the activities will be carried out: independent activities can be carried out simultaneously while dependent activities require the result of a previous activity for their performance.

Gantt chart

A Gantt chart is a tool used to help solve the problem of scheduling activities (i.e. their organisation on a calendar) and represent in pictorial form the duration of each activity, its start and end dates and hence, the total time needed to complete the project. Gantt charts also enable us to monitor the project's progress because they indicate the completed percentage of each activity and detect advances or delays in the initial planning.

Gantt charts have a system of coordinates representing:

- Horizontal axis: time scale, in the appropriate units for the project (usually, days, weeks or months).
- Vertical axis: work packages, activities and subactivities identified in the WBS, whose duration is represented on the horizontal axis.

The main advantage of Gantt charts is that we do not need masses of information to be able to use them; in fact, all that is needed is a rough outline of the project. Hence, they are straightforward to use and particularly effective in the initial planning of the project. However, once the project is underway, particularly if it is very complex, Gantt charts can become confusing.

Critical path method and PERT

To overcome the limitations of Gantt charts, other tools have been developed, such as the critical path method (CPM) and the PERT method.

The critical path of a project is the sequence of activities that generates the maximum accumulated time. It determines the shortest time taken to complete the project if we have all of the necessary resources. To do so, we need to identify all of the activities correctly and know their duration.

To represent the activities and time dependencies, directed graphs are used, whereby each arrow represents an activity identified by its name and duration, so their status changes as the project progresses. Each status is represented by a node between two or more arrows. Thus, some tasks can be conducted at the same time whereas others cannot.

The main difference between CPM and PERT is their time estimates. CPM considers activity times (*m*) to be known exactly and to vary according to the resources allocated to them. PERT, on the other hand, assumes that activity times (*Te*) are determined by a probability distribution generated by the most likely time estimate (*m*), the most optimistic time estimate (*a*) and the most pessimistic time estimate (*b*). Thus, the most pessimistic and most optimistic times give a measure of the uncertainty of each activity.

$$Te = \frac{a + 4m + b}{6}$$

The following steps are taken to calculate the critical path:

1. Calculate *Te* or *m* for each activity, depending on the method used.

2. Calculate the early start dates of each activity (ES) and the last start dates of each activity (LS).

3. Calculate the float of each activity

The float of an activity

The float of an activity is the spare time we have to determine this activity:

- Float of an event: *Hs* = LS of the event – ES of the event.
- Float of an activity: *Ht* = LS of the subsequent event – ES of the previous event – activity duration.

4. Identify the critical path of the project

A critical activity is one whose start and end points cannot be changed without modifying the total duration of the project. Critical activities have no float and the sequence of these critical activities is the critical path. To put it another way, in a critical activity, the early start date will coincide with the latest start date and the earliest finish date will coincide with the latest start date of the activity.

Calculating the ES

The ES of each event is calculated as the maximum duration of the previous activities plus the ES of the previous event. The LS is the last date on which the events can take place without delaying project completion.

See also

To find out more about each of these methods, see the bibliography at the end of the module.

Time management in free software projects

The features of time management in free software projects are theoretically the same as those of any other software project.

In projects developing programmes and applications in which the community of free software developers plays a key role, correct calculation of software development deadlines is essential. To do this, we need to know the background of the community and discuss the future implementation plan. It is also a good idea to get involved in the community and learn about how it works before starting the project.

In migration projects, we need to set aside the right amount of time for training users and introduce a degree of flexibility for the migration of users.

As a result, some free software projects can be accompanied by a degree of uncertainty, so we recommend the use of the PERT technique to characterise the most optimistic and pessimistic scenarios in the project plan.

There are many free applications to create and maintain PERT and Gantt charts.

Free applications

Examples of free applications include GanttProject (http://ganttproject.sourceforge.net/) and OpenWorkbench (http://www.openworkbench.org/).

1.4.3. Integration management

The purpose of integration management is to ensure that the different parts of the project are coordinated correctly. This includes developing the project plan and the project execution plan and tracking any changes that may occur.

Integration management in free software projects

The features of integration management in free software projects are generally the same as those of any other software project, but we need to bear in mind a few points.

By and large, integration in free software based projects has certain advantages over that of proprietary software projects, mainly due to their open source and the use of open standards for interoperability between applications, particularly with those developed outside the project.

In development projects carried out in collaboration with a community not forming part of the project, it is important to make known and discuss the implementation plan both of the project and the community, in order to identify possible incompatibilities that could affect integration.

1.4.4. Cost management

The purpose of cost management is to conclude the project with the budget approved at the start. This means planning the required resources and estimating and monitoring costs.

Cost management in free software projects

Cost management in free software projects differs considerably from that of proprietary software projects.

The main difference lies in licence costs, which are normally non-existent for free software. In contrast, we will need to take into account the costs of services provided by third parties, in accordance with any of the models of business based on free software.

1.4.5. Quality management

The purpose of quality management is to ensure that the project meets the needs for which it was initially designed. We must therefore plan, assure and continuously monitor the quality of the project in accordance with these needs.

Quality management in free software projects

The features of quality management in free software projects are theoretically the same as those of any other software project.

On the one hand, we need to consider quality from the user's point of view, adopting the standards required in each case. And on the other, in the case of projects in which we work with the free software community, whether contributing to an existing project or creating a new one, we need to take into account the quality of the code produced from the point of view of the developer.

We will need to follow the recommendations on programming style, naming conventions, documentation, error logs and formats, languages, etc. With new projects, it is a good idea to circulate these recommendations.

Standard programming styles

Remember that you should follow the programming styles that have already been defined and accepted by the community, such as the Java Code Conventions or Linux C kernel style.

1.4.6. Human resources management

The purpose of human resources management is to employ the individuals participating in the project as efficiently as possible. The activities carried out as part of this management include the organisational plan, hiring of new employees and team development.

Human resources management in free software projects

The features of human resources management in free software projects are theoretically the same as those of any other software project, but you should bear in mind a few points.

Most importantly, you will need to consider the possible participation of the free software community and the effective contribution it may bring to the project. It is generally a good idea to appoint somebody in charge of relations with the free software communities connected to the project.

1.4.7. Communication management

The purpose of communication management is to ensure the correct generation, collection, circulation, storage and elimination of project information within set deadlines.

Communication management in free software projects

The features of communication management in projects based on free software are theoretically the same as those of any other software project. It is very important to ensure the communication and circulation of information within the project, particularly when collaborating with the free software community. If you are not working with the free software community but there is a possibility of releasing the code or the client would like access to the latter, it is also important for the source code to be readable and well documented.

The configuration and correct use of software forges or collaborative development environments (CDEs) will therefore play a key role. Most forges have tools for general project management, bug tracking, forums, mailing lists, etc. Public forges also have these same tools and offer the advantage of more visibility from the free software community.

Software forges

For examples of software forges or collaborative development environments, see the following websites: Gforge, LibreSource and Trac.

Communication tools include mailing lists, IRC channels, blogs, forums and wikis. Relevant decisions made through these tools should be documented correctly and made available to all developers.

Further websites

You can find an example of a public forge at the following website: http://www.sourceforge.net.

We recommend defining rules that should be followed when drafting the documentation, together with the tools that will be used for its automatic generation.

1.4.8. Risk management

The purpose of risk management is to identify, analyse and respond to events that could jeopardise the project plan in the form of delays and increased costs.

These risks must be correctly identified and quantified, and have their appropriate response mechanisms. A risk is always characterised by uncertainty – since the event associated with the risk may or may not occur

– and by loss – because if the event eventually does occur, it will result in negative consequences or losses for the project. Thus, to be able to characterise the risks, we need to evaluate their probability and associated losses correctly.

There are several risk classifications. In this initial approach, we can consider the following three types of risk:

- Management risks, related to problems with scheduling, budgets and the organisation of staff and resources.

- Technical risks, which jeopardise the quality and scheduling of the project and pose obstacles to its development and implementation. The most common technical risks concern potential problems with design, implementation, verification and maintenance. They usually arise from ambiguities in requirements and specifications and the use of outdated or very new technologies.

- Business risks, which raise questions about the feasibility of the project. For example, developing a project for too small a market or one that does not mesh with the company's sales line.

Risk management in free software projects

The features of risk management in free software projects are theoretically the same as those of any other software project.

A classic example of risk in free software projects is the fear and uncertainty – of both the organisation and users – regarding technological change on a new and possibly unfamiliar platform.

It is useful to introduce communication and training methods at the start of the project in order to cancel out any negative effects of the rejection of technological change. It is good practice to earmark a time for presentation, communication and user training that will be continued and built on throughout implementation in order to provide an outline of free software and of the specific applications and tools.

Another classic example of risk in free software projects are the possible legal incompatibilities between free licences for the use of code development and reuse.

It is wise to check at the start of the project that the licences applied to the different parts of the code are coherent among themselves and that the planned development will not generate incompatibilities. This check should

be carried out before every release. It is good practice to keep a licence chart indicating the licence under which each of the software packages is distributed and the individual interactions between them.

1.4.9. Supply management

The purposes of supply management are to ensure that the materials and resources needed to execute the project are available at the right time and in the right place.

Supply management in free software projects

The features of supply management in free software projects are theoretically the same as those of any other software project.

The migration and implementation of a free software system is usually a good time to renew equipment or to modify the structure of the organisation's network. The project plan must therefore include orders of new equipment and materials in order to take into account and prevent possible delays.

2. Free software projects

A project is an organised, structured process of managing resources in order to achieve a specific aim, usually strategic. While, in the first part, we looked at the key aspects of the functional management of resources, this module will focus on the stages that the project needs to complete in order to achieve its aims.

We can generally define seven key stages in projects to implement free software systems:

- Study of the current situation
- Study of the implementation requirements
- Analysis of free software solutions
- Formalisation of the proposal
- Development
- Implementation and migration
- User training, documentation and support

As we can see, these stages are the result of development of the phases described in the first section of this unit and they apply specifically to free software. Nonetheless, the development described is fairly generic and can be applied to other implementation processes.

This section describes the life cycle of the project and outlines the process, its stages and its relationship with the management of the project and the resources allocated to it.

The subsequent seven sections will detail the stages of the project, linking and expanding on the concepts introduced in the first part of this module.

2.1. Life cycle

This first section introduces the main methodological and functional characteristics of the life cycle of the project in order to provide an outline of the process.

The life cycle of the project links the methodical aspects, intrinsic to the development of the implementation stages, to the functional management of the project. Thus, the life cycle guides the execution of the various stages over time and with the available resources.

By and large, the life cycle of a project has two main aims:

- Firstly, it establishes the relationships and dependencies between stages, whether time-based or functional.
- And secondly, it allows us to reduce project risk by dividing its complexity.

See also

To find out more about the management of risks, see the section on risk management in the previous unit.

The project life cycle can be used to monitor the evolution of the stages, the schedule of execution and the financial cost of the project. The management of this cycle is dynamic, so decisions to modify and adapt can be taken as time passes in order to readjust our estimates of the initial parameters on the basis of actual events.

Broadly speaking, the life cycle has four key areas: the project, the stages, execution and the results.

2.1.1. The project

Like any other project, systems implementation projects are designed to achieve a series of aims within a set period and with a given series of resources.

See also

To find out more about project management, see the section on the management of free software projects in the previous unit.

Minimising the time or resources spent on a project will usually minimise the aims that can be met or affect quality, and vice versa. In contrast, minimising project time while maintaining its aims requires us to increase the resources allocated to it. Project management seeks to find the most acceptable balance between these three elements.

In all events, changes in the relationship between these three elements have direct financial repercussions that will need to be assumed in the event that we are updating. The management of the project also has a financial cost from the point at which the project begins (when it is decided to allocate time of one or more staff to take on this management).

The main factors influencing the time and resources required to conclude a project normally refer to the size and complexity of the system to be implemented. In this sense, the features of free software tend to reduce the time and financial costs of the project:

- **Variety of applications**. The maturity of the free software market means the availability of a wide variety of direct implementation products that are reliable, consistent and secure.

- **Licence cost**. Free software is usually obtainable without licence costs and can be downloaded directly from the official website or from other public pools.

- **Source code modification**. The open nature of the source code allows the expansion, modification and adaptation of products that would require a new development to evolve the product if proprietary licence models were applicable.

It is important to note that free software also helps to reduce the overall risk of the project because it provides the freedom to view, use and modify the source code, allowing evaluation and assessment of all aspects of the application in depth.

Additionally, the project can be managed and executed internally or outside the organisation. Broadly speaking, we can differentiate between two main cases:

- ***Insourcing***: this is the case when an organisation develops a project launched as a result of a strategic action. In other words, the organisation's IT department manages and executes the project.

- ***Outsourcing***: this definition applies when an organisation delegates the management and development of a project to an external organisation that manages and executes projects[5]. In other words, the organisation reduces its direct exposure to the development of the project.

(5)One example might be technology consultancies, which carry out projects for other companies.

Thus, the format of the project development will take into account the capacity and experience of the organisation that must assume the development of the project, the associated costs, the implementation schedule and the specialisation of the external organisations present in the project.

Lastly, the project is evaluated in terms of tangible and intangible benefits. Here we can come across cases in which a surcharge for time or financial costs may be feasible in order to obtain the intangible benefits – usually strategic – required by the organisation. For example, improving corporate image with the use and diffusion of free software and the free philosophy.

2.1.2. The stages

The project life cycle is implemented in the form of successive and possibly simultaneous stages. Each stage meets a clear, set aim in a scenario related to the project, with the result that, taken together, these stages meet the aims of the project.

Broadly speaking, a stage can be considered a process that receives inputs and produces certain outputs. In other words, it requires a prior scenario with information about the environment in order to produce certain results.

Hence, a relationship is established between the diverse stages of the project, since each stage achieves part of the global aims. This relationship usually takes two forms:

- **Dependence**: between two stages indicates that a stage requires the result of the execution of the other in order to complete its task. This means that the stages must be executed sequentially over time: the stage generating the results first, followed by the stage that uses these results. For example, the development stage requires the study and analysis of the systems implementation requirements in order to complete its task.

- **Independence**: between two stages indicates that two stages have no direct relationship with each other and no specific prerequisite. This means that the stages can be executed simultaneously, though more resources may be required. For example, the system implementation stage can be executed at the same time as the user training stage.

Moreover, the stages also allow for a distributed execution of the project, that is, one or more stages are assigned to different teams, either internal or external to the organisation (insourcing and outsourcing).

Extreme cases may arise if several stages are assigned to different external organisations. It will all depend on the characteristics of the project, the specialisation of the external organisations and the associated costs.

All this highlights the importance of deliverables between stages. The importance of documenting results in the form of deliverables is threefold:

- Since the documentation of the stage summarises its development and results,
- since the result of the stage is important for the stages that depend on it, and
- since the result of the stage is a result that can be evaluated for the development of the project.

The connotations of the internal or external execution of each stage highlight the importance of deliverables. It should be noted too that their importance is proportional to the complexity and size of the project.

2.1.3. Execution

Execution of the project will begin in accordance with the initial proposed planning, with careful monitoring by the organisation in which the system is to be implemented. By and large, three key parameters need to be monitored:

- **Time**. Time monitoring and management are essential for project monitoring because any change to this parameter will have direct financial consequences. It is also particularly important for the sequencing of the different stages, particularly if they are assigned to different teams.

- **Outsourcing**. Monitoring the outsourcing of the stages or perhaps the entire project is important if we are to ensure that the work and its results meet the aims of the project and the organisation. We need to pay close attention to the monitoring and quality of deliverables and make sure that we follow the schedule correctly.

- **Quality**. The quality control of the tasks completed during execution of the project will have a major impact on the end quality of the implementation. Communication and the transfer of information within the team developing the project and in the organisation itself must also be qualitative in order to guarantee the aims of efficiency and efficacy.

In practice, the execution of the stages can be delayed for a number of reasons that may or may not be linked to the project and its management. Examples of these include a lack of synchronisation in the delivery times of the necessary material, the temporary absence of analysts or programmers and the complexity of a development not initially envisaged. Delays usually have financial consequences.

When a delay occurs, two types of decision can be taken:

- Firstly, the delay can be worked into execution of the stage, concluding it and accepting a delay in all stages that depend on it and, as a result, the overall project.

- Secondly, a project delay may be considered unacceptable and more resources are allocated to one or more stages in order to keep up the pace. Nonetheless, the allocation of more resources does not always lead to production improvements proportional to the resources earmarked.

Delays should not generally have a direct effect on stages executed at the same time as the stage that has been delayed. However, it may be worth considering a schedule adjustment to take into account the delay affecting other stages.

For example, if implementation of the system is postponed because of an excessive delay in the reception of materials, we could consider the option of delaying the user training phase until implementation. This would avoid a gap between user training and application of their knowledge of the newly implemented system.

2.1.4. Results

The results of the life cycle of a project should be directly related to the strategic aims of the project and the organisation. Life cycle in itself is merely a methodical and rigorous way of coming up with a solution for a given problem by dividing the inherent complexity of the project into different stages.

To an extent, life cycle is one way of reducing the overall risk of the project. The execution of the stages as successive fine-tuning periods for solving problems contributes to the gradual adaptation and solution of problems that can be highly complex.

We need to consider the importance of the project management team, which plans and coordinates the project to ensure that it is concluded successfully. Management is a dynamic task that must help to reconcile the differences between planning and the reality of the project during its execution.

The results of a systems implementation project are generally grouped into the following areas:

- **Organisation**. For the organisation, the project needs to meet the expectations of the strategic action from which it derives. We need to stress here the qualitative operation of the system, its integration with the organisation's methodology, user adaptation and competitive improvements to the organisation.

- **System**. The system needs to meet all of the aims and expectations of the organisation and its operation needs to meet the aims of the strategic action of the organisation. The system and its interaction with direct and indirect users must meet these aims qualitatively, offering functional efficiency and efficacy.

- **Users**. One of the aims of the system is to provide technological support to the organisation's operation through its users. The importance of the inclusion of users in the implementation project is strategic, since without their participation in the process and their acceptance of the system,

implementation could prove problematic or unfeasible and have financial repercussions.

- **Documentation**. As with any project, documentation is crucial to the quality of the implemented system for its current integration and future evolution. From deliverable documents between stages to the final documentation or user manuals, all of these materials are crucial to the maintenance and support of the system.

- **Support**. The system must have a support team in place from the start of the project, although it is particularly important in the development, implementation and user training stages. The team needs to guarantee interaction and communication between all those involved in the project during and after implementation, acting as a support team for ongoing training or answering questions and solving problems.

In all events, a systems implementation project must allow the organisation and its users to evolve towards new strategic challenges. Creating a climate of confidence and acceptance of change is essential if it is to achieve its aims.

Acceptance of change

This process is usually called **change management** and covers all of the aspects and procedures that enable us to manage and solve any problems and misgivings with the implementation of a new system in the organisation, especially those implemented in free software.

2.2. Study of the current situation

This section will define systems analysis and describe its main characteristics and special features. It will detail the various phases of the study, the main factors influencing its development and the results that the analysis should produce.

See also

For more information about the relationship between the implementation project and the organisation's strategy, see the sections on the organisation's strategic plan and the origin of systems implementation in the first module.

Systems analysis is a chiefly theoretical form of investigation designed to provide a clear and accurate view of the status of the organisation's system within the scope of the project, based on the strategic action from which it is derived.

Systems analysis covers two complementary aspects:

- Part of the analysis involves the technological application of the case study, with a qualitative evaluation of the system from a methodological and procedural perspective.

Case study

Case studies are a scientific method that allows us to explore an object or circumstance in depth through the use of empirical strategies in order to understand the object of study. It is usually used in the initial exploration and in combination with other techniques such as the quantitative approach (statistics related).

- And part of the analysis involves the study of the compliance or competence of the organisation's system, with a quantitative evaluation of the system from a functional and technological perspective.

The theoretical implications of the investigation reveal the importance of a methodical, rigorous and exhaustive approach. Errors in appreciation at this stage can generate problems later on and possibly raise doubts over continuing with the project because of biases affecting the project, the current system and the organisation's strategy, with the ensuing financial repercussions[6].

(6)It is not only necessary to take into account the direct financial costs of time spent on the project, but also all indirect costs, such as the cost of abandoning a project that has been started and the cost of the lost opportunity.

Although this section describes the features of the initial study for a system that has already been implemented, this structure is equally applicable to newly implemented projects if we transfer the object of study to the scope of the organisation, the current and past markets, future technological trends and similar projects begun previously.

This first stage of the project may also bear no direct relationship with free software, since the aim is to analyse and evaluate the implemented system or the current market, regardless of the form of implantation or trend, respectively.

Broadly speaking, we can divide systems analysis into three main phases: identification of the system, preparation or development of the case study and the final evaluation.

2.2.1. Identification of the system

The purpose of identifying the system is to define the object, scope and aims of the study. The definition of these parameters is directly related to the strategic action on which the project is based and must allow us to set up the scenario for evaluation.

It is important to remember that a system already in place is not simply a set of technological parts; it is also a series of features, methods and procedures that have a direct impact on users and the organisation in general.

Hence, the study scope must cover the technological aspects of the implementation, the features currently covered by the system, the procedures and methods of action deriving from its interaction with the organisation's operation and the impact on the use of the direct and indirect users of the system.

We need to determine two key aspects of these parameters:

- Firstly, we must determine the different sources of information that will allow us to obtain data for the subsequent analysis of the system.

- And secondly, we must identify the quantitative or qualitative nature of the data we will obtain from the sources of information, since data extraction techniques vary widely.

See also

The following section on case study development describes the main differences between techniques for obtaining sources of information.

Quantitative and qualitative data

Quantitative data are numerical variables that quantify characteristics or attributes. For example, the number of active users in the system per unit of time.

Qualitative data are variables that differentiate between characteristics or attributes but do not quantify them. For example, the colour combination of an application's user interface.

The result of this phase is a working document indicating the object, scope and aims of the study, along with a list of the data that must be obtained and the source of this information.

2.2.2. Case study development

This phase is used essentially to collate all of the important study data indicated in the system identification phase.

In practice, information can be gathered from a range of sources: historical documents, detailed interviews, results of technological audits, performance counter tools, the documentation of previous projects, technological specifications and even issue reports.

As we collect data, we may come across interesting aspects that were not considered in the system identification phase. In all events, data collection needs to be rigorous and meet the criteria on structure and organisation.

Nonetheless, we can differentiate between two generic groups in data collection:

- **Quantitative data**. This type of data is usually collected directly from technological media. The implemented system may have counters for performance, transactions, capacity, volume, etc., which can generate interesting statistical results, as in the case of time or cost units.

- **Qualitative data**. This type of data is usually collected from written documents, meetings or staff interviews. The procedure for obtaining information from interviews and meetings is crucial here, because they must be painstakingly prepared and conducted if we are to obtain quality information.

It is very important to use a methodical process to obtain both quantitative and qualitative data, since the system is a support tool for staff and the organisation itself. All information is beneficial when it comes to assessing and evaluating the system.

This phase usually marks the beginning of the development of the hardware and software inventory and the current system's network diagram. Besides being useful for determining the current status, it can also be efficient and help with the planning of a possible system migration.

See also

You can find out more about hardware and software inventories and network diagrams in sections 2.7.3 and 2.7.4 of this module.

The end format of the case is usually an investigative report in which all of the aspects dealt with above are organised and evaluated. The report must back up the data and results it describes, relating them to one another and the definition of the project and seeking out possible relationships of dependence or independence.

Depending on the type of information described, it may be convenient to use statistical results, tables, graphs and charts, and generally anything that will help with the description, understanding and evaluation of the data included in the report.

One of the most common tools used for the presentation of executive summaries are SWOT[7] analyses, which present the main conclusions of the study of the current system from a strategic point of view. If the features of the project require, we can produce SWOT tables by classifying the various features of the system according to the result of their evaluation, such as if the current system's hardware is a weakness for carrying out the strategic action successfully.

(7)SWOT is an acronym for Strengths, Weaknesses, Opportunities, and Threats.

2.2.3. Final evaluation

The final evaluation of the system is the first control point of the project and its aim is to determine the feasibility of the current system in the light of the organisation's strategic actions and hence, evaluate the need to continue with the project.

Generally speaking, there are four large groups of features that we need to consider:

- **Operating**. These relate to the functional interaction of the users with the implemented system, ergonomics, performance, efficiency, efficacy and usefulness.

- **Organisational**. These concern the methods and procedures generated by the implemented system, together with their benefits and disadvantages for the organisation.

- **Functional**. These relate to the efficiency and efficacy of the tasks carried out by the implemented system, in addition to scope, reliability, performance and malfunctions.

- **Legal and financial**. These relate to the cost of the implemented system and its legalisation, covering aspects such as maintenance, licensing and system administration.

Evaluation of the system can bring us to three main groups of conclusions:

- **The system is feasible**. The study and evaluation conclude that the current system is ready to take on the strategic actions of the organisation. These conclusions are usually drawn in cases where a study has been conducted to find out the status of a large and/or complex system whose evolution could be difficult to gauge on a superficial level.
  If the current system obtains a positive evaluation, the implementation project must be cancelled because there are no indications that a new implementation is required.

- **The system is partially feasible**. The study and evaluation conclude that the current system needs minor updates before it can take on the strategic actions of the organisation. These changes usually involve updating or changing a small series of elements, such as replacing devices or updating software.
  A partially positive evaluation of the current system requires continuing with the implementation project but it is convenient to revise the aims and scope in order to adapt them to the needs detected.

- **The system is unfeasible**. The study and evaluation conclude that the current system cannot assume the strategic actions of the organisation. This type of conclusion is usually drawn in cases of migration from older systems that are unreliable or perform poorly and hence need to be completely updated.
  A negative evaluation of the current system tells us that we should continue with the project to implement a new system. It may be necessary to revise the aims and scope of the project because replacing the current system may require more resources than those initially envisaged.

Both the report on the analysis and the final evaluation of the current system are submitted to the organisation by the project monitoring committee. The final decision on whether or not to continue with the project is usually made by the management of the organisation.

The result of this stage is twofold:

- Firstly, we obtain a comprehensive report on the current status of the system, highlighting the main features of the system from the point of view of the organisation's strategy.

- And secondly, the organisation makes a decision as to whether or not to continue with the project and any actions required to adapt the system to the latter's strategy.

2.3. Study of the implementation requirements

This section will define the study of systems implementation requirements and describe its main characteristics and special features. It will detail the various phases of the study, the main factors influencing its development and the results that it should produce.

The study of the system requirements is a process requiring a methodological analysis of the problems that need to be solved.

There are two main aims to the requirements study:

- **To define the implementation**. Requirements studies allow us to detail all the features that the system to be implemented must have and permit. To a certain extent, they define the specific, functional aims of implementation.

- **To reduce risks**. Requirements studies also allow us to reduce the risk of the project and its management by specifying and progressively fine-tuning the characteristics of the solution to be implemented.

Requirements studies for systems implementation projects usually require a great deal of effort for two main reasons:

- Firstly, because it can be difficult to specify and methodologically structure the ideas and expectations of users and the managers of the organisation regarding the new system, bearing in mind that user requirements can evolve over time.

- And secondly, because they are crucial to the subsequent development of the project, since errors in appreciation made in this phase and detected in later stages will affect the timing and financial aspects of the project or generate unplanned extra costs that could jeopardise completion of the project[8].

(8)Errors made during the system design stages detected and solved in the development stages have a possible cost ratio of one to ten.

A requirement is a feature that the new system needs to have. In other words, it is an attribute that must allow the system to meet the set aims. Requirements are usually written as text but can take the form of tables and charts if these help to clarify and specify the aim.

In general, we can define four different types of requirement:

- **Strategic policy**. Requirements linked to the organisation's strategic policy cover general aspects of the project, its management, outcome, or the approach to adopt. These include corporate ethics and image or the traditional attributes of the organisation.

- **Methods and procedures**. The implementation of a new system is usually a good time to update and improve the methods and processes of the organisation and/or system. Besides a thorough revision of current procedures, we will also need to bear in mind the specification of future methods and procedures resulting from new aims or features that the new system will need to encompass.

- **Operation of the system**. The operating requirements of the system derive from the interaction of users with the system. We will need to distinguish between functional and non-functional requirements: functional requirements correspond to specific actions that the system will need to execute, whereas non-functional requirements correspond to limitations or restrictions while executing actions, which allow the actions to be linked to the methods and procedures.

- **Key factors and priorities**. Most systems have a specific number of basic elements that form part of the core of the system. In new implementations, we need to give preference or priority to the components considered essential for the operation of the system and the organisation; other components not considered essential can be given a lower priority.

The requirement collection stage is not exclusive to free software because free software systems implementation must meet the same requirements and aims as any other type of implementation.

Broadly speaking, we can divide the study of system requirements into five main phases: identification and definition, specification and structuring, verification, validation and the final evaluation.

2.3.1. Identification and definition

This first phase of the requirements study pinpoints and defines the problems that need to be solved, indicates the project typology and determines the main sources of information for data collection.

See also

See Section 1.2 of the first module to find out about the different types of systems implementation projects.

The requirements study stage uses documentary information from the analysis of the current status stage, so part of the task of pinpointing the problems has already been completed.

We will need to identify the resources and elements from the organisation that are directly or indirectly involved and define the scope of the problems for the organisation and for those resources and elements that are directly and indirectly involved, whether human, material, functional, organisational, procedural or technological. All these elements will usually allow us to recover vital information for defining the new system, which we can use, together with the report on the current status, to establish a corpus of knowledge for making decisions.

The direction in which the market is heading is also important when it comes to defining requirements. Familiarity with the features of similar systems, organisations from the same sector, recent innovations and the future trends of the project theme can all be helpful when it comes to specifying, proposing, evaluating and understanding the requests of users and organisation managers.

The result of this first phase is usually presented in a working document with a detailed definition of the aim and scope of the project, a list of the elements to consider, sources of internal information on the organisation and a list of elements or market trends that may contain relevant information for the project.

2.3.2. Specification and structuring

The aim of the specification and structuring phase is to collect relevant data on all of the elements indicated above and to organise them using methodological criteria to create a reliable corpus of knowledge that accurately represents the reality.

This phase is based on the working document produced in the previous phase, providing an initial outline of the elements and sources of information that may contain relevant information for the project. Nonetheless, the practical development of the study may lead us to consider new sources of information and new aspects of relevance to the project that were not taken into account in the previous phase. Deadlines permitting, we should try to investigate all of the details that may appear.

We can identify two tasks in the development of this phase:

- **Collection and specification**. This task attempts to resolve the functionalities of the system. In other words, what the system has to do, not how it has to do it. Some information sources tend to focus on how actions should be done instead of determining the specific features of the task itself.

- **Structuring and organisation**. This involves organising data methodically and comprehensibly. It may be useful to prioritise certain requirements over others, in accordance with the elements of the system required for its operation.

As with the study of the current status, data collection may respond to quantitative or qualitative criteria:

- **Quantitative data**. These usually come from technological media and can be bulk processed in order to obtain statistical results to verify and justify qualitative data and to model and extrapolate results to new functionalities.

- **Qualitative data**. These are usually taken from interviews and meetings with those involved and provide us with functional and non-functional results regarding the system and the procedures and methods affected by the project.

Systems implementation requirements relate to the basic elements of the whole system, such as the hardware, software, infrastructure, staff, procedures, functionalities and even languages, documentation, formats and standards.

Requirements are usually presented as text, organised according to the characteristics of the project, system and the special features of the requirements themselves. Tables, graphs and charts can also be used to improve the definition, arguments and evaluation of the ideas covered.

The result of this phase usually takes the form of a working document that describes all of the requirements, organised, structured and reasoned in accordance with the ideas of the project. The clarity and precision of this document is crucial for the entire project, since all subsequent development and the organisation's acceptance of the terms of the final implementation both depend on it.

2.3.3. Verification

The requirements verification phase evaluates the requirements contained and described in the previous phases and assesses them in the context of the system's coherence and the aims of the organisation.

This phase uses the working document from the previous phase, which methodically organises the requirements obtained in the study.

The formal verification of the requirements can be split into two main processes:

- **Technological analysis**. The technological analysis of requirements is a process aimed at analysing and concluding that all of the requirements complement one another and form a logical system whose implementation is feasible.
  Antagonistic requirements resulting from the diversity of information sources are usually detected in this phase. This conflict can be resolved by checking the other characteristics covered by the requirements and/or validating the options with those directly involved or with the organisation.

- **Functional analysis**. The functional analysis of the requirements is a process aimed at analysing and concluding that the system deriving from implementation of the requirements meets the requirements and aims of the project and the organisation.
  In this phase, we can detect inaccurate assumptions in the requirements, which can contradict the aims of the project. The conflict can be resolved by a review and consultation of the problematic aspects with those directly involved and/or with the organisation.

Requirements verification is a fundamentally technological and methodological process that analyses the coherence and feasibility of the future implementation and introduction of the system defined in the requirements.

It can be interesting to have a number of people participate in the review of the requirements. Different perspectives can help pinpoint and solve any errors or shortcomings more effectively.

The review can also stimulate the appearance of new issues or situations not previously considered, which may generate a loop between the collection of requirements and their verification until convergence and coherence have been established in the results (a similar process to top-down methodology).

Top-down methodology

Top-down methodology takes the general definition of a problem and creates a loop by specifying and refining each item until a sufficient level of detail is considered to have been reached. The result is usually displayed as a process tree.

The result of this phase is that the working document with the requirements, created in the previous phase, is updated. This updating has allowed a detailed review of the concepts of implementation and the identification and resolution of any errors in the initial requirements.

2.3.4. Validation

The requirements validation phase is designed for reaching an agreement on the requirements of implementation of the new system – following the study – with the organisation. Agreeing on the requirements is crucial to the contractual formalisation of the implementation of the system.

This phase requires the active participation of the organisation, which must conduct a thorough analysis of the requirements proposal resulting from the previous phases. The transfer of information between the two groups is essential because the characteristics of the implementation of the system depend on the understanding of the requirements study.

The internal working document containing the requirements may not be entirely appropriate for presentation to the project's monitoring committee. In this case, it may be necessary to create new documents or presentations to transmit the information in a clearer and more effective way, using charts and diagrams.

The organisation's validation of the requirements may conclude with new revisions of the requirements. These revisions or adjustments usually involve the addition of new functionalities to the system by the organisation that were not initially planned in the project.

Nonetheless, these revisions cannot continue indefinitely because they could have a direct impact on the feasibility of the project and its implementation schedule, not to mention the financial implications of the increase in the number of requirements to implement.

The result of this phase is closely related to the final evaluation phase of the systems requirements study. It usually generates the last revision of the working document with the requirements of the new system, agreed and validated by the organisation and the team that conducted the requirements study.

2.3.5. Final evaluation

The final evaluation of the requirements study is the second control point of the project, the aim of which is to determine the feasibility of the implementation project in the light of the requirements of the new system and hence, the need to continue with the project.

This phase is closely related to the validation of the requirements, since the main working document represents the last definition of the requirements for implementation of the system, agreed on by the organisation. The validation and final evaluation phases may be merged for making decisions regarding the need to continue with the project.

The feasibility of the system requirements is evaluated by taking into account their suitability for the current system, the organisation and the strategic action that launched the project. The requirements study is the first formal step towards the new system and early estimates of the volume and cost of the changes.

In a way, the evaluation of the requirements is similar to the evaluation of the current status that we looked at in the first section, which bases its assessment on financial, technological, functional and legal aspects. We can also evaluate other aspects relating to the strategy of the organisation, such as the latter's capacity to evolve, the intangible benefits of the change, management quality and corporate ethics.

The evaluation of the proposed requirements for the new system can lead to three main groups of conclusions:

- **The project is unfeasible**. The evaluation of the requirements of the new system by the organisation concludes that implementation of the new system is unfeasible and the project is abandoned.
  There may be a number of reasons for reaching this conclusion, some of which could even be outside the scope of the project. This outcome is usually related to economics, financing, competition or the abandoning of the strategy that launched the project.

- **The project is partially feasible**. The evaluation of the requirements of the new system by the organisation concludes that it will be carried out in part, meaning that only certain elements of the new system will be implemented.
  This outcome is usually related to economics and financing. In some cases, partial implementation can be carried out progressively or in stages[9]. In these cases, it is important to introduce guarantees to ensure cohesion between the different implementations over time.

[9] Some organisations prefer to spread the burden of the investment of a new implementation over several accounting years.

- **The project is feasible**. The organisation has made a positive evaluation of the requirements of the new system and the project will continue without major limitations or restrictions that could affect the basic definition of the project.

The result of this stage is twofold:

- Firstly, we obtain a full report on the requirements of the new system to implement, validated by the team and the organisation.

- And secondly, the organisation makes a decision as to whether or not to continue with the project and on the scope of the implementation of the new system.

2.4. Analysis of free software solutions

This section will define the analysis of solutions for the implementation project and describe its main characteristics and special features. It will detail the various phases of the analysis, the factors influencing its development and the results that this stage should generate.

Solutions analysis is a process whereby the various technological options available are analysed methodically and rigorously in line with the project requirements.

This analysis has three complementary aims:

- **To know the market**. The analysis of solutions allows us to study and evaluate the current market situation in terms of the definition, scope and aims of the implementation project. The variety of solutions currently available means that an deep, careful analysis is required.

- **To adapt the solution**. The analysis allows us to select the solutions that best fit the problems of the project and the implementation of the system. Moreover, the open nature of the source code inherent to free software allows for the fine-tuning and final adaptation of the chosen solutions in the form of derivative products.

- **To reduce risks**. The analysis of solutions allows us to reduce the risk of the project and the implementation of the system because the study allows us to adapt the solution and control the main repercussions of its implementation.

With free software, the solutions analysis stage takes place in a scenario limited by two main conditions:

- **Project typology**. The project typology, the characteristics and the scope of the system to be implemented will determine the scope of the search for and analysis of possible solutions, together with the feasibility of the various proposals.

- **Analysis of requirements**. The analysis of system requirements must determine the functional and operating behaviour of the solution in terms of the project aims.

As we could expect based on what has gone before, this stage uses the documents on the definition, aims and scope of the project and on the system requirements agreed with the organisation. The report on the evaluation of the current situation can also be useful here because it indicates the characteristics, special features, and conclusions on the viability of the current system. All of this must help us to focus on the characteristics of the analysis.

This stage focuses mainly on the strength and diversification of the market's current supply of free software. The search for and selection of free solutions that meet the requirements is essential for the implementation project. Hence, our analysis must reflect the competitive attributes of free software in comparison to private solutions, with a special emphasis on freedoms of use and source code adaptation.

Broadly speaking, the free software solutions analysis stage can be divided into three main phases: the search for solutions, the analysis and assessment of candidates, and the final evaluation.

2.4.1. Search for solutions

This first search phase is aimed at identifying solutions whose implementation in the framework of the project is feasible. It is used to make an initial selection of systems with a similar aim to those of the project.

Free software is a valid and viable option for systems implementation since it offers a wide range of freedoms of use, operation, access and modification of source code, and licensing of derivatives.

These features not only allow them to be used freely by organisations and private individuals, they also permit independence from proprietary providers, savings on licences and royalties, learning from the original source code, monitoring of obsolescence with guaranteed product maintenance, quality and reliability, and guaranteed security, privacy, interoperability and software convergence.

Free software implementation can traditionally be divided into two main areas:

- Implementation in infrastructure services[10]
  Since it first emerged, free software has maintained a very close relationship with systems architecture and network services. It is now the undisputed leader of certain sectors[11], ahead of proprietary software.

- Implementation for home users or clients[12].
  As a result of various past initiatives, free software has now embarked on a new path and spread to the environment of the end user, entering into competition with the *de facto* standards of the home environment.

The market now offers a wide variety of free software systems in very diverse fields. Most important projects have their own websites to promote the knowledge, diffusion, downloading of and collaboration with the project[13].

There are also public pools[14] allowing the creation and development of and collaboration on new free software projects, together with the downloading of the resulting applications. These pools often act as a launch pad for community projects.

It should be remembered that there may be no one solution for the implementation project, whether due to scope or the specialised nature of the aims. In these cases, we need to categorise our search for solutions into more generic typologies, so that a number of specialist systems can be put together to form a joint solution to the project requirements[15].

The search for solutions should generate a document with the results of the research that covers the following aspects:

- **Definition and identification of the search for solutions**. Here we indicate the characteristics used to direct our search, explaining their relationship with the project definition, aims and scope, and the definition of the agreed requirements.

- **List of solutions**. The document must contain a list of the solutions we have come across in the search process, briefly defining the solution and its relationship with the project aims for each.

- **Summary of technical features of the solutions**. The summary of technical features of each solution must tell us about the main characteristics of the system, such as the definition of the project, the pools in which it is found, the monitoring and maintenance of the product, the languages it supports, the community collaboration, its end

(10) For instance, local network servers form part of the basic infrastructure services of the organisation.

(11) For example, the implementation of *Apache Web Server* on web servers is superior to that of other environments.

(12) For example, the Ubuntu distribution competes directly with *Microsoft*'s operating systems, debunking many myths about difficulties with the installation, management or use of GNU/Linux environments.

(13) Examples include the Ubuntu distribution (www.ubuntu.com), the office suite OpenOffice.org (www.openoffice.org) and the browsing suite Mozilla (www.mozilla.org).

(14) For example, SourceForge.net (sourceforge.net).

(15) For example, if we are looking for a solution for the management of a web database, one option might be to search for a separate operating system, web server, database manager and programming tool that can work together to offer the required functionalities. This is the case of the LAMP environment (*Linux*, *Apache*, *MySQL* and *PHP*), which is currently very popular.

licence and other requirements specific to the product, such as ergonomics of use, execution requirements and the main functionalities.

If there is no single solution for the problems of the project and we need to split it into different individual solutions, the document can be organised into categories by typology (for example, operating systems or office suites) or by function (such as a database server with a web interface for access and programming).

A SWOT table is also sometimes included for each of the options found. This table is very important in meetings and executive decision-making, and its aim is to identify and summarise the main characteristics and specific advantages and disadvantages of adoption of these options.

2.4.2. Analysis and assessment of candidates

The aim of the analysis and assessment of candidates phase is to identify the solutions most suited to the project and the organisation and to the special features of the development, implementation and migration.

This phase uses the document from the previous phase, which lists the most suitable current solutions for the project and details their main characteristics. The purpose is to methodically and carefully select the best candidates for implementation from among the diverse alternatives identified.

A series of technological parameters directly related to the project, organisation and implementation process are usually analysed, considered and evaluated. The individual assessments produce an organised classification that will determine how well suited each solution is to the implementation project.

We can generally use the following parameters to assess and evaluate a solution:

- **Project and organisation**. We must assess its adaptation to the project aims, the organisation, the definition of methods and procedures, the ethics and traditional standards of the organisation and the strategic action that launched the project.

- **System and interoperability**. Here we need to assess how well it meets the needs of the system and confirm its guaranteed operation with current or future hardware, software and network resources, the feasibility of the methods and procedures, implementation of the strategic action, interoperability with existing systems and standards in the organisation and the possibilities for future expansion and evolution.

- **Functionality and ergonomics**. We need to assess how well it meets the functional and operating needs of the project and the organisation, and confirm the ergonomics of its use and implementation of the methods and procedures with guaranteed operation and that it will meet the expectations of the users and the organisation.

- **Efficiency and performance**. We need to assess whether it can guarantee efficient operation at all times, the use and performance of allocated resources and maintenance of a suitable balance between resources and performance over time, to allow evolution.

- **Efficacy and reliability**. We need to assess whether it guarantees performance of the functions set down in the project aims, maintains functional compliance with the aims at all times, preserves the balance between resources and compliance over time and that allows evolution.

- **Implementation and migration**. We need to consider how well suited it is to the system implementation process and confirm that it allows efficient and effective migration from an earlier system, minimises the consequences of impact on the everyday operation of the organisation and that it guarantees tools for the support, training and adaptation of users and any other systems currently in place.

- **Maintenance and management**. We must assess whether it will equip the system with management and configuration tools adapted to the project aims, minimise physical and logical maintenance operations, guarantee a balance between operation and maintenance over time, and allow evolution.

- **Support and commitment**. We must evaluate whether the support and monitoring of its creators for the system and its users are guaranteed, and whether it contributes to the community commitment to evolution, future improvements, problem solving and generally all aspects that could cause the system to become obsolete.

- **Licences**. We must evaluate whether it guarantees compliance with the legislation in force, establishes a scenario for use and operation, allows a clear and effective distinction to be made concerning actions that can be performed with the system and specifies the licences for any products derived from it.

- **Economics**. We need to evaluate how it adapts to the economy of the organisation, the planned project financing, the costs of management, maintenance and its efficient and effective operation throughout the expected life of the system and the costs of training and educating the users of the system.

As we can see from the above list, the parameters seek to assess candidates in different areas (technological, strategic, operating, financial, etc.), in order to detail, analyse and evaluate the impact of implementation on the organisation.

The result of this phase is a document that numerically classifies and considers the various candidates (possibly by category), organising the solutions according to how well they adapt to the project and the organisation.

Given that different solutions could obtain the same technical score, it may be necessary to conduct another study on these alternatives that, despite differing in their characteristics, have similar degrees of adaptation to the project. In these cases, it may be useful to draw up a SWOT table for each solution in order to reveal the differences that could help us to make our final choice.

2.4.3. Final evaluation

The final evaluation of the analysis of free software solutions is the third control point of the project and its purpose is to determine the development of the implementation project, i.e. the form of the project and its final implementation.

This phase uses the documents from the previous phase, which classify the existing solutions based on how suited they are to the project, and the summaries of technical features and/or SWOT tables analysing them in detail.

This phase has two main aims:

- **Final selection of candidates**. Although the previous phase indirectly proposes the most appropriate solutions for the project and implementation (by scoring and classification), it may also be necessary to evaluate the overall integration of all of the solutions.

- **Development of the project**. The choice of the solutions meeting the project requirements will have direct consequences on its development, such as its cost in terms of time and financial resources.

Remember that the components of the system will not be separate from one another, so the best solutions will be those that generate a stable and operative system besides meeting the requirements of the project.

In some cases, it may be useful to carry out integration tests to determine the quality and performance of the cohesion of the diverse elements, since the integration of solutions that have obtained high scores in their respective categories does not necessarily guarantee an overall performance of the same level[16].

(16)For example, if the connector between the two systems is inefficient or limited due to compatibility with other systems.

The choice of solutions will generate adaptations in the project, particularly during the development phase. Overall, we can consider three different types of implementation:

- **Direct implementation**. Solutions most suited to the needs of the project and the organisation can be implemented directly, either because they are of general use or because the project aims meet the specific needs of a large group.
  Nonetheless, differences between the requirements of the project and the functionalities of the solution do not automatically rule out adoption.

Solutions for direct implementation

Examples of direct implementation solutions include operating systems, office automation packages, e-mail clients or web browsers.

- **Evolution of the solution**. The solutions most suited to the requirements of the project and the organisation can only reveal significant differences at certain points. Their adoption requires an evolution of the source code to adapt the possible shortcomings of the original solution in the event of problems with the system.
  The freedoms allowed by free software make it a valid and viable option in practice, while the proprietary software alternative may require a completely new development, with the ensuing financial implications.

- **New development**. There are no solutions that adapt sufficiently to the problems of the project and the organisation, so a completely new development is required to solve them.
  These cases may occur in systems with very specialised requirements, such as robotics. In exceptional cases, proprietary solutions may not exist in the usual marketing channels.
  In all events, the freedom for studying, analysing and reusing the source code of free software allows us to improve the quality, efficiency and efficacy of the new development and cut associated costs.

The evaluation of free software solutions can lead us to three main types of conclusion:

- **Feasible direct implementation project**. The project is feasible and the implementation requirements can be met by existing solutions. Any differences between the solutions to be implemented and the problems in the project are not insurmountable[17].

(17)For example, MS Word can be replaced with OpenOffice.org and functional differences can be dealt with by user training.

- **Feasible project subject to conditions**. The project is feasible but the implementation requires the use of a process to create or adapt solutions.

Time and money implications will be dictated by the scope of the process, which will be at its largest when a completely new development is required.

- **The project is unfeasible**. Although it is not usual to abandon the project at this stage, the organisation can sometimes decide not to continue with it. For example, if financing for a specific development is not available or if the organisation changes strategy.

The result of this stage is twofold:

- On the one hand, we obtain free software solutions adapted to the requirements of the project and the organisation.

- And on the other, we obtain a preliminary evaluation of the costs involved at the development stage of the project (direct implementation or the need for further development).

2.5. Formalisation of the proposal

This section will define the formalisation of the project implementation proposal and describe its main characteristics and special features. It will detail the various phases of the process, the main factors influencing its development and the results that this stage should generate.

The formalisation of the proposal is the stage of the project in which we detail, specify and link all of the results obtained in the early methodological phases of the project – called *design stages* – with the aim of putting forward a formal solution to the problems associated with implementing systems in the organisation.

This stage has three specific aims:

- **To characterise the project**. Formalisation of the proposal collates, clarifies and lists all of the results from the initial design stages of systems implementation. The result of this stage should serve as a guide for the subsequent development of the implementation, hence it is a key part of the project.

- **To obtain validation from the organisation**. The formalisation of the project also serves to present the implementation project to the organisation, which must then evaluate it. The resulting evaluation will determine the project's future, viability, development and execution.

- **To reduce risks**. Formalising the proposal allows us to reduce the risk of the project and implementation of the system, and delimit the scenario of the project's evolution by detailing solutions and the development of the implementation.

This stage is related to the functional management of the project because it allows us to detail aspects of project planning such as management of time, resources and project costs using the information obtained from the results of the previous stages.

Overall, the proposal needs to cover two main aspects:

- **Methodological aspects**. Methodological aspects refer to the results of the analysis and systems implementation design stages that we have seen in this unit. Specifically, these are analysis of the current system, analysis of requirements and analysis of free software solutions. The aim here is to detail the characteristics of the system to be implemented using a rigorous and methodical process.

- **Management aspects**. Management aspects cover the special features of the operation and execution of the project, such as scheduling, financial planning or others relating to the human resources needed to conclude the project. The aim is to detail the project execution parameters in order to obtain a clear vision of the requirements needed to meet the objectives.

See also

See these and other aspects of project management in the "Management of free software projects" section of the first unit.

From the preceding paragraphs, we can conclude that communication and collaboration with the organisation is important if we are to convey all of the information regarding the project correctly. By and large, we can identify two main aspects of the information that need to be conveyed:

- **The project**. Understanding of the project, the results obtained from the analysis and the solution most suited to the project characteristics. The relationship between the project proposals and the strategic action of the organisation must be clear.

- **Free software**. Understanding of the use of free software, its philosophical basis and the features of its licences, together with the special features of the solutions proposed and the benefits obtained by the organisation.

Broadly speaking, we can divide the formalisation of the proposal into four main phases: drafting, design, presentation and the final evaluation.

2.5.1. Drafting of the proposal

This first phase of the formalisation of the proposal is designed to recover all of the information that can be incorporated into the proposed solution. Here, we should obtain the documents resulting from the various stages of study, analysis and project planning.

One of the main aims of this first phase is to generate advanced results from the initial data. In other words, we should create summaries, diagrams, tables, charts and graphs to clarify the concepts in the technical documents drawn up throughout the project.

Another result of this phase is the updating of the project planning using the information contained in diverse analytical documents. The most important aspects for the organisation are usually:

- **Solution to be implemented**. The definition of the solution to be implemented, the aims and scope of the project, the strategic problems solved and the systems to be implemented are all important.

- **Time costs**. Time costs are related to the scheduling of resources and of implementation or migration itself.

- **Financial costs**. Financial costs are related to the financial repercussions of implementing the system in the organisation (materials, resources, time, training, etc.) and the project management (project management team). Remember that you will need to take into account the financial costs of the project from the outset[18].

See also

See the "Time management" section for more information about the time costs of the project.

See also

See the "Cost management" section for more information on the financial costs of the project.

It may also be useful to prepare the following aspects:

- Seek out classical misconceptions about free software, particularly concerning the proposed solutions, and provide arguments for and against. The aim of this exercise is to provide a working basis for justifying and establishing the grounds for the proposal in the event of a possible reticence to change[19] observed in its presentation.

- Obtain and present information on free software licence typologies in an attempt to prove certain assumptions wrong and to transmit the free philosophy and its relationship with the proposed solution in the right way.

- Draw up SWOT tables to represent and justify situations where it has been necessary to make decisions based on the study and evaluation of different

(18) As mentioned in the "Resources of a systems implementation project" section, the project starts when it is allocated resources for the first time.

(19) FUD (*fear, uncertainty* and *doubt*).

options. In these cases, it may be useful to evaluate and consider each of the alternatives presented.

This phase generates a series of documents with results on two main aspects:

- Firstly, a set of references to technical analysis documents from the design stages and diverse advanced results and information relating to the project and free software.

- And secondly, an update to the project management planning, which includes all aspects deriving from the analysis and design stages, shaping continuance of the project.

2.5.2. Design of the proposal

The design phase of the proposal is used to draw up all documents and presentations that must be submitted to the organisation and which summarise the work carried out in the study and analysis stages of the systems implementation project.

This phase uses the documents from the previous phase, with references to technical documents, information on the project and free software, advanced results and updates to scheduling and financial planning. The aim is to structure, organise and present all available information so that it can serve as a guide for the subsequent development of the project.

By and large, we can generate two main types of document:

- **Reports and schedules**: these are documents presenting the results of a deep study. Their aim is to present the contents in a structured and methodical fashion using accuracy, order and rigour so that they can be used as a guide or handbook for the project.

- **Presentations**: these are documents that reveal the general lines of the study. Their aim is to describe the main concepts of the study in an orderly, synthetic and graphical way, so that they can be used as the summary of the project.

Reports should only be presented after editing and formatting and they should adopt a formal and precise language. Reports normally contain the following:

- **Executive summary**: this is a brief summary that positions the systems implementation project and its contents in terms of time, space, its aims and its solutions.

- **Introduction**: this describes the strategic position of the organisation and the need to carry out a systems implementation project in specific circumstances.

- **Analysis of the current situation**: this should include a summary of the study, an analysis and the conclusions on the organisation's current system in relation to its strategy.

- **Definition, aims and scope**: this describes the systems implementation project, the general concepts of the proposed solution, justification of the use of free software and its suitability for the organisation's strategy.

- **Architecture of the system, infrastructure and technology**: this describes the general and functional architecture of the system, the infrastructure it requires, the technology and standards it uses, and the end licences. Integration tests for the different elements may also be included.

- **Human and material resources**: here we describe the human and material resources needed for the everyday operation of the system in the organisation.

- **Implementation and maintenance**: this describes the implementation methodology used, the features of migration, user training, adaptation of the organisation and the necessary maintenance of the installation over time (usually over a period of five years).

- **Scheduling**: this details the monitoring and relationship between the stages of the project over time, taking into account the resources allocated at any one time. It usually takes the form of a chart in which the stages are plotted against the time axis.

- **Financial planning**: this describes the assessment of the cost of conducting the project, breaking down the cost of implementation into various items (e.g. cost of project management, necessary human resources, material expenses, software development and adaptation, infrastructure installation, etc.).

- **Synergies with the free software community and other projects and bodies**: depending on the characteristics of the organisation, we can include a section describing the relationship between the solutions used and the free philosophy sector, the use of similar projects and relations with other bodies or with the free software community.

- **Conclusions:** we need to highlight the project features that meet the strategic needs of the organisation and the benefits obtained from the use of free software.

- **Technical appendices**: these can come in useful for the subsequent development of the project. For example, for the analysis of the system requirements.

We obtain two documents from this phase:

- The report or project plan, which is presented as technical documentation of the project.
- The presentation, which will be used at the executive meeting in the subsequent phase.

2.5.3. Presentation of the proposal

In this phase, we present the proposal to the organisation, usually during an executive meeting with its management body. The purpose is to inform the organisation of the final results of the analytical stages so that the organisation can evaluate the proposed solution and the monitoring of the project, and consider its continuity.

This phase is usually implemented by presenting all of the aspects of the project in the form of charts and summaries, using the presentation document from the previous phase. In addition to the presentation, we must deliver the project plan with the results of the design stages.

It is particularly important in this phase to focus on the communication and transfer of information, highlighting the relationship between the proposal and the strategic aims of the organisation. Though not the norm, we may be asked to reformulate some aspects of the proposal, which will, in any case, need to be fairly minor in quantitative and qualitative scope.

The committee in charge of validating the implementation of the system may be reticent to the use of free software, so it may be wise to justify the proposal by questioning certain preconceived ideas, explaining the advantages and disadvantages of its use and even discussing the free philosophy in general and licence models. The document drawn up in the design phase of the proposal on issues related to the project and free software may prove useful here.

This phase is closely related to the subsequent phase, the final evaluation, since decisions on whether or not to continue with the project are often made at presentation meetings.

2.5.4. Final evaluation

The final evaluation phase of the formalisation of the proposal is the fourth control point of the project, designed to evaluate the work carried out in the design stages of implementation and the proposed project solution. It must also resolve the feasibility and continuity of the project for the development and system implementation stages.

This phase is closely related to the previous one because the presentation of the project plan and exchange of information between the organisation and the team that came up with the design is essential if we are to take into account all of the implications of the proposed solution. The proposal presentation and validation phases may be merged for making decisions regarding the continuity of the project.

The importance of this control point is twofold:

- Firstly, because the formalisation of a proposal closes the theoretical and conceptual cycle of the analysis and design of the system and its associated implementation project. The subsequent stages will be mainly practical.

- And secondly, because the organisation will finally determine whether or not the project will be concluded and meet the aims of the strategic action, implementing the series of changes considered in the proposal.

Hence, as explained in the previous sections, the organisation must evaluate the aspects of the project it considers to suit its strategy; for example, the characteristics of the solution to be implemented and the financial costs of the project and implementation of the system.

It might also be a good time to analyse, evaluate and adapt certain specific details of the implementation of the system in the organisation, considering and defining certain aspects of change management, implementation scheduling, pilot tests and user training.

The organisation's evaluation of the proposal can end with one of three main types of decision:

- **Feasible project**. The organisation accepts the proposed solution without making any major changes to its specifications. Any adaptations relate to the implementation schedule or the inclusion of certain financial items, for example.

- **Feasible project subject to conditions**. The organisation considers certain adaptations to the proposed solution to be necessary. These changes may or may not be minor, but part of the proposal will have to be drawn up again if the project is to be given the go-ahead. The changes may be related

to the tools implemented or to the partial execution of the project, for example.

- **Project unfeasible**. The organisation considers the project unfeasible and abandons the implementation of the system. Although this does not usually occur at this stage in the project, a drastic change in organisation strategy or the inability to secure financing to conclude the project are possible reasons for abandoning implementation.

The result of this stage is twofold:

- Firstly, we obtain the project plan, which is agreed and validated by the organisation.
- And secondly, the organisation makes a decision as to whether or not to continue with the project implementation.

2.6. Development

This section will define the development stage of the implementation project and describe its main characteristics and special features. It will detail the various phases of the process, the main factors influencing its development and the results that this stage should generate.

The development of the system is the stage in which the solutions described in the project plan are implemented, together with all of the requirements for its start-up. The purpose is primarily to adapt, collect and produce all of the necessary elements in order to perform implementation with the utmost guarantees of efficiency and efficacy, and to minimise intervention time.

This stage uses documents from the previous stage, mainly the project plan and analysis of the system requirements to be implemented. These documents are essential for the preparation, structuring and organisation of the development of all components needed to implement the project.

This stage has two specific aims:

- **To implement the solution**. The purpose of development is to implement the solution set out in the project plan and the study of requirements, regardless of the form of the development type.

- **To reduce risks**. Development also attempts to reduce the overall risk of the project by preparing, specifying and building the solution without directly affecting the operation of the organisation.

See also

The "Requirements Verification" section of this module describes three project typologies: direct implementation, the evolution of existing tools and the development of a new solution.

In general, development needs to cover two main aspects:

- **Methodological aspects**. Methodological aspects relate to the project typology, so the development stage is directly related to the purpose of implementation. In other words, the development methodology to be applied will depend on the aim of the project.

See also

For more information on project typologies based on aims, see Section 1.2.2 "Classification by project aim".

- **Management aspects**. Management aspects concern the people involved in the development, i.e. those who implement the tasks set down in the project plan. Broadly speaking, we can have internal development (insourcing) or external development (outsourcing).

See also

For more details about internal and external development, see the "Resources of a systems implementation project" section of the first unit.

In general, we can divide the development stage into three main phases: the allocation of resources, the configuration or development of the software and the final evaluation, although the individual existence of these stages will depend largely on the project typology.

The allocation of resources and software development phases can be executed at the same time, depending on the project type. It is also possible for these two phases to overlap rather than being simultaneous, perhaps because of differences in the delivery dates of the various resources.

We also need to remember that the development stage is closely related to the subsequent stage: implementation or system migration. In this case, some tasks may overlap or be carried out at the same time in order to cut short the project schedule, though this could have financial consequences that need to be taken into account, caused by aspects such as the need for more human and material resources.

2.6.1. Allocation of resources

The aim of this first allocation of resources phase is to select and obtain all of the necessary resources to implement the system in the organisation, in order to obtain all of the elements directly required for the implementation and operation of the new system.

One of the main aims of this first phase is to equip the organisation with the necessary infrastructure for system start-up, i.e. all of the material resources, organisations and configurations that will allow the new system to become operative[20]. Therefore, if the project includes the total or partial migration of the system, it may be useful to link this phase to the migration planning phase.

[(20)]Examples include installing a local area network, enlisting a local technician or purchasing switches, computers and servers.

See also

To find out more about migration planning, see the sections on "Migration strategies", "Hardware and software inventories" and "Network and structure diagrams".

Material resources are usually allocated after evaluating the characteristics of provider proposals based on a set of specific conditions. This set of conditions must be created using the system requirements to ensure that it adapts and integrates with the rest of the project.

Evaluation of the diverse proposals is specific to the project, system and organisation. Broadly speaking, we can assess the following:

- **System and interoperability**: we need to evaluate each element alone and its adaptation and integration with the system, project and organisation.

- **Functionality and ergonomics**: we must assess the ease of handling and configuration of each element and the knowledge required for its fine-tuning and everyday use.

- **Efficiency and performance**: we need to assess functional and operating performance in line with the needs of the system.

- **Efficacy and reliability**: we must evaluate the reliability and coverage of the aims in relation to the system requirements.

- **Implementation and migration**: we should evaluate the ease of introducing the element into the organisation and the tools it offers for migration from the current solution.

- **Maintenance, support and guarantee**: we must evaluate the operation and maintenance needs of the element, and the support and guarantees offered by providers.

- **Economics**: we need to evaluate the cost of the element in terms of its promised results and the advantages and disadvantages of the impact on the organisation's operation.

Implementation of the project may also require the allocation of human resources to the organisation, for either direct or indirect system handling. Since the project stems from a strategic action within the organisation, a change in methods and procedures may require a restructuring of the staff involved.

The biggest changes in this regard would take place in new organisations or existing ones that do not use computer equipment in their day-to-day operation. The demand for technological profiles goes hand in hand with the level of technological implementation, the type of project and the characteristics of the organisation.

In all events, the selection of human resources will have to be integrated into the typical recruitment methodology of the organisation but we should stress the need for the candidate profile to correspond to the technological requirements of the organisation.

2.6.2. Software configuration and/or development

The aim of the software development phase is to implement all adaptations required by the free software solution in order to adapt it to the project plan. This phase can also include the development of tools to help with data or file conversion for systems migration.

The main aim of this phase is to ensure that the free software solution matches the project requirements, with the adaptation and encoding of all necessary changes, taking advantage of the open nature of the source code and the freedom to develop it.

The length of this phase will depend directly on the changes that need to be made to the proposal. Generally, we can distinguish between three different cases:

- **Direct implementation**. When the software is to be implemented directly, the development scenario is limited to adaptation of the configuration files or software parameters.
  For example, the operating system configuration (language, network access, etc.) or the parameters of office automation tools (templates, language, etc.). This case requires minimal investments in time and financial resources in comparison to the other two.

- **Software evolution**. Software evolution encompasses the extension or adaptation of the source code of one or more free software solutions to adapt its operation to that of the organisation.
  The importance of the results means that we must proceed with the usual rigour of software engineering, establishing an adequate life cycle for each modification; it could also include software re-engineering.
  One example might be the introduction of additional calculations in accounting management software or the modification of the downloading manager of a web browser.

- **New development**. New code is developed when no solutions can be found to meet the specific needs of the organisation. Nonetheless, it can still prove useful to study and reuse the source code of open applications to save time and ensure reliability.
  New developments are the most laborious in terms of time and cost. This means that we must proceed using software engineering methodologies adapted to the project. For example, the creation of industrial design or architectural or electronic device control software.

See also

For more information on free software production, see the subject on "Software engineering in free software environments".

Hence, any software development project will be guided by its specific life cycle and will establish the relevant control and quality milestones, while evaluating monitoring and results using the most appropriate methodology.

Regardless of the development format, in this phase, it is useful to generate the documentation for the software adaptation process with a list and description of all details subject to modifications, using the same rigour as for new developments. In these circumstances, we need to take into account the end licence for the evolving solution, since the freedom of the source code is usually inherited from the original solution.

It may also be useful in this phase to develop the materials for training the system users and to adapt the support manuals of the free software solutions, since we will now have all of the necessary elements to perform this activity. In these cases, we also need to take into account the characteristics of the licences for the manuals we wish to edit.

Support manuals

Frequently asked questions (*FAQs*) or the standard manuals for completing specific tasks (*How to...*) which, although highly technical, can still be very useful.

2.6.3. Final evaluation

The final evaluation phase of the development stage is the fifth control point of the project and its aim is to ensure that the development meets the requirements of the project plan. It is also designed to reduce the overall risk of the project by validating the viability and adaptation of development to the organisation's strategy.

The evaluation of this stage usually depends on the typology of the development:

- **Resources, installations and infrastructures**: these are evaluated according to whether or not they meet the aims of the design and the provider's proposal. Delivery deadlines are very important here as they have repercussions on the scheduling of other stages.

- **Direct software implementation**: we must evaluate whether or not the aims are met by the proposed adaptation of the configuration and operation. The main evaluation is based on operating tests.

- **Software evolution or development**: the complexity of the evaluation is proportional to the scope of the changes made. The evaluation resulting from the software engineering process is normally taken into account.

This phase is closely related to the implementation and migration of the system since, depending on the project type, we can implement tailored solutions to conclude the development. It can also be related to user training and pilot tests because it can be useful to find out user opinions in order to fine-tune certain aspects of the development[21].

[21]For example, the graphical interface, system response or the characteristics of the procedures that it implements.

The importance of this control point is twofold:

- Firstly, because it revises, evaluates and assesses the qualitative development of the solutions and their adaptation to the strategic requirements of the organisation.

- And secondly, because it supposes the completion of the process of creating and adapting the solution, and the start of the definitive implementation and integration of the system in the organisation.

The development evaluation phase is generally a good time to introduce users to the features of the system, and the situation can be regarded as the start of the training and adaptation of users to the new environment. These actions can take place as part of the management of the change that the organisation needs to carry out in order to overcome any misgivings or fears among users[22].

(22)FUD (*fear, uncertainty and doubt*).

The final evaluation of the development stage can end in one of three main types of decision:

- **System implementation is feasible**. The evaluation determines that the system meets the requirements of the organisation and that all modifications of the original solution have been developed and integrated correctly. Implementation of the system in the organisation may begin.

- **System implementation is partially feasible**. The evaluation determines that the system partially meets the project requirements on the control date. This situation is normally due to production delays, which can occur as a result of unexpected events, such as the lack of material resources, long delivery delays or the lack of human resources for production.
  In all events, the feasibility of the project is not affected, but we will need to take into account the fact that implementation will be delayed.

- **System implementation is unfeasible**. Although it is rare for the development of a solution to be unfeasible, there may be exceptional factors affecting the decision that could not be solved during this stage. This type of problem is normally associated with factors outside the scope of the project, such as sudden changes in strategy or a lack of funds.
  Abandoning the project at this stage will have serious financial, functional and moral implications for the organisation and will make it difficult to take any further action in the future.

The result of this stage is twofold:

- Firstly, we obtain the system to be implemented (tested and integrated), which will meet the project requirements and the strategic needs of the organisation.

- And secondly, we will have assessed the feasibility of implementation and made any adjustments to scheduling as a result of the evaluation of pilot tests.

2.7. Implementation and migration

This section details the technical aspects of migration to free software systems and their implementation.

Most implementation projects are also migration projects because they start off with a scenario in which there is a computer system based on proprietary software already in production. Implementation from scratch takes place in new organisations that look to free software solutions to start up their first system or in organisations that currently have no computer system, which is rarely the case.

Implementation from scratch is always much more straightforward than migration in terms of the technical problems that the project will need to solve, mainly because there will not be any compatibility problems with existing systems. Nonetheless, we do need to bear in mind that the users of the new system will usually be familiar with proprietary operating systems and applications, so planning the training is just as important in migration projects.

This section will refer exclusively to migration projects because they are the most difficult and because projects implemented from scratch can be considered a subset of migration projects with the particular feature that they allow greater freedom to decide on the final scenario. This freedom comes with the condition that more attention must be paid to non-functional aspects of the system, such as correct dimensioning.

We will begin by describing the different types of migration project and the most crucial aspects of the planning. We will then offer advice and guidance for executing the migration and evaluating its results. And finally, we will conduct a detailed study of all the services involved in the migration project and the most popular free software solutions for each.

2.7.1. Types of migration

We can carry out a range of possible types of migration to free software systems in organisations: aims-based, or depending on the elements of the system that will be migrated; and scope-based, or according to the number of elements that will be migrated.

Aims-based:

- **Migration of services and servers**: this affects the servers of the organisation, the applications they run and the services they carry out, such as authentication or printing services, among others. In this case, there is no change to the clients' applications, so we only need to plan for system administrator training and not for end users. These are among the easiest migrations to carry out. The servers, which use GNU/Linux, operating systems from the BSD[23] family or other free systems tend to be more reliable and offer better performance, which will increase the productivity of the organisation, both by systems administrators and end users (lower server response time).

- **Migration of users and clients**: this affects the client machines of users and the applications running on them. In this case, we will need to train and accompany the end users, who are generally less receptive to the use of new applications and will be most affected by the change, which could cause a temporary loss of productivity.

- **Migration of applications**: this only affects some of the applications running on client machines or servers, the operating system of which does not necessarily have to be GNU/Linux or any other free operating system. More often than not, the operating systems continue to be proprietary. It is sometimes a preliminary step towards migration of the operating system. These are fairly straightforward migrations, such as those to OpenOffice.org or MySQL Community Server.

Scope-based:

- **Complete migration**: this is the result of combining the migration both of servers and client machines. This type of migration must be planned in such a way as to ensure that clients are not left without access to the services provided by the servers at any time. Hence, the first step is usually the total or partial migration of servers, followed by the migration of client machines.

- **Partial migration**: this is the result of a combination of the partial migration of the servers or part of the clients, which means that there will still be machines based on proprietary software in the system. One common scenario is that where a single system contains clients based on free software and clients based on proprietary software, whose configuration will depend on the needs or preferences of the end users.

- **Migration based on virtualisation**: this can be viewed as a type of partial migration where we migrate servers and client machines at the same time as we continue to install and run proprietary software applications that could not be included in the migration, whether because there are no free

[23] OpenBSD, FreeBSD or NetBSD.

See also

The "Network and structure diagrams" section details the migration features for each of these services.

software equivalents or for other reasons. With virtualisation, we can start up a proprietary operating system on a free operating system and use it as normal with proprietary software applications.

We also need to remember that while the typical migration scenario is one where we switch from a proprietary operating system to GNU/Linux, there are other possible combinations, such as:

- From a proprietary operating system to a free operating system, such as those of the BSD family.
- From a free operating system to another free operating system.

The OpenBSD operating system

The OpenBSD operating system is renowned for the quality of its security mechanisms and its integrated cryptography, making it ideal for servers or other machines whose integrity could be compromised.

2.7.2. Migration strategies

As with any project, correct planning is essential if we are to migrate successfully to a free software system. There are as many ways to plan as there are projects and they will all be valid to an extent if they meet the requirements and features of our migration scenario. Nonetheless, depending on our migration planning, we can extrapolate four main migration strategies:

- **Single-step migration**: this involves carrying out the entire migration process in a short space of time, in a single day or during public holidays if possible. The strategy requires identifying and defining all of the tasks that need to be performed, since a mistake in any of these could leave the entire system inoperable, with the subsequent risk of delays and user rejection. This is the most economical strategy and is usually used for small systems with few clients and a single server, as is the case of small companies.

- **Pilot migration**: the first step is to migrate a small part of the system, which will be used for practising and evaluating the success of the migration before we implement the rest of the system. The pilot system usually consists of a number of servers and client machines, but it can also be a single server and a single client machine. Although it is very important to plan correctly, this strategy offers greater flexibility for modifying the migration approach and correcting possible problems. The drawback to the strategy is that it requires far more resources and is therefore usually used in organisations with medium-sized or large systems.

- **Group migration**: this involves defining user groups based on their functional characteristics and carrying out migration progressively with each of these groups. One of the main advantages of this strategy is that it enables us to minimise risks and learn from each migration. Moreover, since migration only affects part of the system, we can reduce productivity losses. The disadvantage is that we often need to keep the previous systems

running while we set up the free software system. It is usually a good time to renew hardware when carrying out group migration, and vice versa.

- **User migration**: this is similar to group migration, the difference being that only one user is migrated each time. As a result, the strategy requires very few resources but is unfeasible in large and medium-sized organisations, where migration would take too long due to the high number of users. It could, however, prove useful for the migration of critical systems and users that require special monitoring.

These strategies are not mutually exclusive and several of them can be used in any one project. For example, in an organisation that has some smaller or less important departments, single-step migration can be carried out in these, while pilot migration can be carried out in others before moving on to complete implementation. Similarly, group migration can be seen as a combination of single-step and pilot migration.

2.7.3. Hardware and software inventories

In order to plan migration, we need to identify the hardware and software available in the initial situation that we wish to maintain after migration. As a result, we must carry out a detailed inventory of both hardware and software.

The hardware inventory should include all machines available for migration including those that have been withdrawn, since we may be able to use some of these again.

Hardware can be grouped into the following categories:

- **Hardware with full GNU/Linux support**: this includes hardware with out-of-the-box support in the Linux kernel or free drivers, and hardware for which we need to use proprietary drivers, either directly or using adaptors. Most hardware has good support on GNU/Linux, so it will fall into this category. We will describe each of these situations in detail later on.

- **Hardware with limited GNU/Linux support**: this includes hardware that only works with older versions of the Linux kernel not used in the more recent distributions of GNU/Linux, hardware that runs with very old drivers whose maintenance has expired, and hardware with free drivers that have functional limitations in comparison to the proprietary drivers[24].

(24) For example, graphics adapters with 3D acceleration whose free drivers are only available in 2D.

GNU/Linux distribution

The production of a GNU/Linux distribution involves checking that all the packages included in the distribution are compatible with one another and, most importantly,

with the kernel version. Thus, there will always be a gap of some months between the date of the appearance of the distribution and that of its kernel, which is older.

- **Hardware with no GNU/Linux support**: this includes hardware that does not work on GNU/Linux. In actual fact, there is very little hardware without GNU/Linux support but when this is the case, it is either because the hardware is very new and the relevant drivers have not yet been developed, or because the hardware is very old and incompatible with the newer versions of the Linux kernel or because the hardware is dependent on a specific operating system and cannot therefore be used with GNU/Linux[25].

(25) In this case, however, a virtualisation solution could still be possible.

We can also classify hardware with GNU/Linux support by the type of support, as follows:

- **Hardware with out-of-the-box GNU/Linux support**: most equipment and devices have adequate support in the recent GNU/Linux distributions so there is no need for external drivers. It is easy to find lists of hardware with GNU/Linux support on the Internet.

Website

You can find out which hardware has GNU/Linux support at: http://hardware4linux.info/.

- **Hardware with free driver support**: although they do not have out-of-the-box GNU/Linux support, many devices work correctly with drivers maintained by the free software community. The package managers included with the GNU/Linux distributions often propose the installation of these drivers when they detect the hardware.

Website

NDISwrapper is a free software project enabling the use of wireless network cards with GNU/Linux through the use of Windows drivers.

- **Hardware with proprietary driver support**: the free software community does not maintain drivers for this type of device so we will need to use proprietary drivers, often supplied by the manufacturer. This is usually the case of hardware with very specific functionalities, such as graphics accelerators. Nonetheless, support for this type of device is gradually increasing and the manufacturers themselves often open up their drivers.

- **Hardware with adaptor support**: this hardware is supported by proprietary operating systems but not by GNU/Linux. Fortunately, tools called adaptors are available so that we can use the drivers of these proprietary operating systems in GNU/Linux.

Once we have classified the hardware correctly, we will obtain a clear idea of the available resources and the need to purchase any new equipment. Moreover, as we mentioned earlier, the hardware requirements of GNU/Linux systems are considerably lower than those of proprietary operating systems, so obsolete machines can be reused for services such as printing or e-mail.

Once we have completed the hardware inventory, we will need to do the same with the software. We will have to identify all of the proprietary software applications used in the system before migration and the best free software applications for replacing them.

There are many lists on the Internet indicating equivalent proprietary and free applications. However, a detailed study of the functionalities of each application is sometimes required in order to select the best candidate from the free software options.

Website

The SourcePYME project publishes a fairly extensive and up-to-date list of applications and services (http://www.sourcepyme.org/?q=node/13). There are also many resources available in English.

There are many options for the more common applications, such as office automation, but there is often one application or package of applications that stands out above the rest. There are communities of developers and users for applications with more specific uses and it can often be a good idea to ask their advice or even get involved.

See also

The "Evaluation of migration" section contains the available alternatives for migrating the core services of an organisation.

If there are no free software applications or projects that meet the organisation's needs, it may consider developing a new application and opening it up with a free licence, provided that it has the resources to do so. The benefits are clear: the potential contribution of external developers and users and increased visibility for the organisation.

2.7.4. Network and structure diagrams

This section describes two essential elements of systems migration: the network diagram, which illustrates the connectivity between the diverse elements of the system, and the structural diagram, which indicates its physical location.

Thus, once we know which hardware and software will be affected by migration, we can represent the system on a network diagram. This diagram must contain the following elements:

Website

An excellent free software application for creating network diagrams as well as other types of diagram, is Dia (http://live.gnome.org/Dia).

- **Servers**. Indicate the name of the equipment, together with the main services offered by each server.

- **Client or user equipment**. Indicate the name of the equipment and the network devices exposed to the other systems.

- **Printers**. Indicate the name of the printer and the print server or client computer on which it depends.

- **Other network equipment**. Indicate the main equipment forming the system network enabling connectivity between the various machines. For example, hubs, routers, switches and wireless *access points*.

- **Connectivity between elements**. Indicate the wired and wireless network connections between the various elements of the system. Indicate the organisation's local network connection to external networks such as the Internet, virtual private networks (VPN) and virtual organisations (VO).

It is very important for each machine to be uniquely identified in the network diagram.

First of all, we need to draw up a network diagram illustrating the status of the system before migration. This diagram will be used to study the possibilities for optimising the current network, such as alleviating server bottlenecks or connecting certain local printers to a central print server. We will also decide which new equipment and network elements will be introduced in the system as new servers, which old equipment can work with GNU/Linux or discuss the implementation of a wireless network.

With these elements we can create a network diagram for the system after migration. This diagram will be essential for defining the planning and strategy of the migration and will serve as a guide during implementation. Thus, it is a good idea to keep the network diagram up to date, ensuring that it offers a true picture of the system status.

As explained at the beginning of the section, the structural diagram reflects the physical location of the equipment inside the organisation; in other words, it tells us what equipment there is in each room.

As we did with the network diagram, we will create a structural diagram to illustrate the status of the system before migration, which can then be used to decide on the location of the equipment after migration.

One of the most common consequences of migration is the introduction of servers for a small number of services, sometimes even exclusively. As a result, the servers are usually grouped together in the same physical location (usually a server room) that meets the specific requirements for climate control, power supply and accessibility, among others. Another example is the connection of printers (thus far local) to a print server, which can be located in the same room.

Although structural diagrams are not very important in small organisations with just ten or so machines, in bigger migration scenarios, it is essential to know the location of each piece of equipment and network component.

2.7.5. Execution of migration

When carrying out any migration, regardless of the strategy adopted, there is a series of technical tasks that almost always crop up again and again: equipment installation, data migration, production of backup files, emulation of applications, etc. Besides these technical tasks, it is important to have a management plan in place to deal with any risks that may appear during migration.

This section will look briefly at each of these tasks and offer guidelines for their completion:

- **Equipment installation**. Automatic equipment installation tools are available for easy installation and configuration of more than one in a short space of time.

One such tool is SystemImager, which automates the installation of clones of the GNU/Linux system installed on an initial piece of equipment. SystemImager also allows us to distribute new applications or data on the system equipment and make changes to the configuration or install system updates on networks with GNU/Linux equipment. However, if the hardware of the equipment is not identical, they may need to be manually configured.

Website

You can find out more about SystemImager at http://wiki.systemimager.org/.

SystemImager

SystemImager allows you to save a clone of a GNU/Linux system in production before making changes to the system, which means that you can revert to the original situation if need be.

- **Migration of user data**.

The names and addresses of users are usually stored in directory services, normally accessible through the standard LDAP protocol, which facilitates the migration of these data at system level.

However, in the case of the applications that use these data, such as e-mail clients or groupware, different data schema are often used to structure the information. As a result, the data are rarely interoperable and external programmes must be used to synchronise the data between applications.

See also

See the section on "Hardware and software inventories" for more details on the migration of directory services.

- **Making backups**.

Backup is the term generally used to refer to the copying of data to allow a system to be restored after loss of information. The implementation of GNU/Linux systems often involves the formatting and partitioning of the equipment involved in the migration, so we need to make backups of the existing data in order to restore it later on the new system.

If the organisation has an up-to-date backup mechanism, one option is to use this to recover all the information that needs to be copied to the new equipment.

If the organisation has no backup mechanism, we can set up a storage service exclusively for the storage of the data that needs to be migrated. Another option is to implement the storage service envisaged in the project plan first and then provide access to the system users so that they can store their data before the user equipment is migrated. In both cases, the participation of users is fundamental.

Once migration is complete, we will need to set up an incremental backup mechanism[26]. As a general rule, the original system and the backup must be as independent as possible so that an error in one will not affect the other.

(26)There are many solutions available, including RSync (http://samba.anu.edu.au/rsync) and Amanda (http://amanda.org).

- **Emulation of applications and virtualisation**.

Performance of the software inventory will serve to determine which applications cannot run natively on GNU/Linux and cannot be replaced by an equivalent free application. If these applications are essential and we need to continue using them, we have two possible solutions: emulation or virtualisation.

Wine, the most popular free solution

Wine is the most popular free solution for running native Windows applications on a GNU/Linux system. Although Wine (http://www.winehq.org/) is usually referred to as an emulator, it is more correct to say that Wine provides a layer of compatibility for Windows applications. Wine is in fact an acronym of Wine Is Not an Emulator.

Wine does not need to install a Windows partition but it can be useful to have some native Windows libraries in some cases. Applications that run with Wine can access the file system, network and printing services in a completely transparent way. The Wine website contains information on the supported applications and their level of functionality.

For applications that do not run correctly with Wine, it is possible to run them on a virtualised operating system. As explained earlier, virtualisation allows us to run one operating system over another. In this case, we would run the application on a virtual Windows system on a GNU/Linux system. The most popular free virtualisation solutions are QEMU, Xen and KVM. In all events, virtualisation should always be considered as a last resort because we will have to carry on using and paying for proprietary licences and because it is a big consumer of system resources.

- **Risk management**.

Migrating to a free software system is not without risks, so it is important to draw up a management plan and keep to it for the duration of the project.

See also

See the section on "Risk management" for more details on how to draft a risk management plan.

The risks and their relative importance will depend on the migration scenario and the features of the organisation. For example, some organisations may consider it a priority to guarantee the security and integrity of certain confidential data, so this contingency will need to be provided for and a plan drawn up to solve it in the event that it does occur.

As a general rule, we suggest that you keep the migration process reversible until you have fully verified the new system, i.e. that you will be able to return to the starting point in the unlikely event that migration fails or proves unfeasible.

2.7.6. Evaluation of the migration

In any migration project, it is essential to evaluate both the end system and the migration process. This evaluation can be done once we have completed the migration, but it can also be carried out during the process, if it is not conceived as a single-step process.

Hence, the project plan must include a series of clear indicators. These indicators may include some of the following, which refer to the operating system, servers, applications and users:

- **System indicators**. Have the reliability, performance and security of the system increased since migration? How has the real (as opposed to estimated) cost of system maintenance changed? Have new services been introduced into the system? How do the administrators see the new system? Has the number of problems with system services fallen since migration?

- **Operating system indicators**. How many machines have been migrated to the new system? Does all of the equipment work properly? Is all of the hardware supported by the new system? How often have virtualisation solutions been required?

- **Application indicators**. For how many existing applications has an equivalent application in free software been found and implemented? What functionalities have been gained and lost with regard to the original applications? How many applications run through emulation or virtualisation? How many applications was it necessary to modify? How many applications have had to be developed from scratch?

- **User indicators**. How many users have been migrated to the new system? What is their opinion of the functional and non-functional aspects of the new system and the new applications? How has their productivity varied

in the short and long term? Has the number of user problems fallen since the new operating system was installed?

2.7.7. Migration of the services of a system

Most organisations have a series of basic services that must be paid special attention when planning and executing migration:

- File system
- Printing service
- Directory and authentication services
- Network service
- System management and administration
- Web servers
- Databases
- Desktop environments and office automation applications
- Corporate applications

This section will describe the main characteristics of these services and indicate the most popular free software solutions. There is generally more than one alternative so the final choice will depend on the characteristics and requirements of each scenario.

Clarification

Although most of the free software solutions described in this section are in the mature stages of development and used in many scenarios, technology evolves constantly, so it is a good idea to visit the websites of the projects to obtain more recent technical information and research other solutions that could improve on existing ones.

The importance of each of these services will also vary according to the characteristics of the organisation. Some of these services may not exist in the initial situation and will not therefore be included in the migration.

Nonetheless, migration is an excellent opportunity for analysing and revising the current status of the system and designing an architecture that meets both the current and long-term needs of the organisation. Hence, we need to consider the inclusion of new services not present in the original system.

File system

We can be faced with two situations when migrating the file system, depending on whether all or only a number of the clients are being migrated:

Migration of the storage system server but not the client server

In this case, the most popular solution is Samba, a free protocol implementation used in Microsoft Windows shared file systems for Unix systems that allows computers with GNU/Linux to act as servers or clients in Windows networks.

Migration of both the storage system and client servers

In this case, we will usually use NFS or OpenAFS.

NFS allows us to access remote files in the same network as if they were local files. NFS is included in the GNU/Linux operating system by default. Similarly, OpenAFS[27] is a distributed file system generally used in clusters and distributed computing scenarios.

(27)OpenAFS is a free implementation of a file system originally developed by Carnegie Mellon University that also influenced the design of NFS.

The choice of one or the other (or the choice of another system) will depend on the migration requirements. It is possible to use NFS or OpenAFS in networks that include Windows and GNU/Linux clients.

For the migration of servers operating with GNU/Linux, there are several file systems but the most widely known are Ext3 and XFS. Their functionalities include journaling, assignment of quotas and access privileges based on ACL (Access Control List) by file and directory.

When migrating file systems, we need to pay attention to the mapping of the Windows ACLs to Posix ACLs, since we can lose granularity here. In practice, this does not usually occur because organisations usually do not make full use of the granularity permitted by Windows ACLs.

Printing service

Printing is one of the most common sources of problems in organisations, generally because printers are installed with no planning, which leads to numerous several technical issues and, very often, a waste of resources (paper, ink, electricity). Migration to a free software system is actually a good opportunity to optimise the printing service.

Of the available free software solutions, CUPS is the print server used by most GNU/Linux distributions and is in fact the best option for almost all migration scenarios. One of its main advantages is that it provides us with a print service both for GNU/Linux clients and Windows clients, because it implements *Internet Printing Protocol* or (IPP).

IPP is a printing standard for both LAN and WAN networks that supports communication between clients and servers, between different servers and between the selected server and printer. It is supported by all modern printers.

Before carrying out migration, we need to check the support and drivers for each printer.

Directory and authentication services

The purpose of a directory service is to ensure that certain information is available to all network users. This information is usually composed of objects organised hierarchically, originating from a root object. The most common access protocol is LDAP.

LDAP (*Lightweight Directory Access Protocol*)

LDAP initially referred only to the access protocol but it has come to mean the combination of the database containing the information and the protocol for accessing it.

For example, a fairly common use for a directory service is to store system user accounts together with their privileges so that all the system applications and services can access it to obtain this type of information, which should be complete given its nature.

Thus, a directory service needs to offer the following functionalities:

- The available information must be modified and organised into a hierarchical structure.
- Use of a standard data schema to ensure compatibility and interoperability with as many applications as possible.
- User authentication and guaranteed interoperability with other authentication services.
- Administration of access rights to the information in the directory service.
- Secure transfer of information between clients and the directory service.

For authentication with a directory service, the most common free software solution is a combination of OpenLDAP and Samba, where the latter serves as a database of user accounts and OpenLDAP acts as a directory service. There are many applications compatible with LDAP, including the office automation package OpenOffice.org.

The GNU/Linux system offers various LDAP tools[28] for modifying the information stored in the directory service and there are also web-based graphical interfaces available for the administration of users and groups.

(28) These include ldapsearch, ldapad and ldapmodify.

A directory and authentication service based on OpenLDAP and Samba will also allow the simultaneous use of Windows and Linux clients. In fact, OpenLDAP also acts as part of the authentication service and as an integration tool in mixed scenarios with GNU/Linux and Windows clients. If we are carrying out a complete migration to GNU/Linux, authentication is also possible with Kerberos. Kerberos is an authentication protocol that allows two computers to reveal their identities to one another securely.

Network services

The entire TCP/IP network infrastructure (DNS, DHCP, NTP, router connection, filtering, VPN) can easily be implemented with free software solutions, due mainly to the fact that all Internet protocols are open standards, both in their definition and in their implementations.

Note

For more information about open standards, see Appendix II of this module.

One aspect to consider when migrating network services is the use of open standards, even when they are not needed (as in the case of small local networks) as this does away with the need for specific modifications from hardware manufacturers, which can eventually cause compatibility problems with other systems when new services are implemented and even a dependency on the manufacturer.

The network services include the following:

- **DNS (*domain name system*)**
  The free software implementation of reference is BIND (Berkeley Internet Name Domain), currently maintained by the Internet Systems Consortium (ISC). BIND is the most popular DNS server on the Internet. Its latest version is BIND 9, which includes DNSSEC (DNS Security Extensions), TSIG (Transaction Signature), DNS notification, *nsupdate* and *Ipv6*, among other functionalities. It is available on all GNU/Linux systems.

Internet Systems Consortium

The Internet Systems Consortium is the non-profit organisation that succeeded the Internet Software Consortium, also known as ISC.

- **DHCP (*dynamic host configuration protocol*)**
  The implementation of reference in free software is `dhcpd` , also now maintained by the ISC. `dhcpd` allows the administration of individual clients and group configurations for classes and subnets. Moreover, `dhcpd` offers load balance functionalities and high availability. It is available on all GNU/Linux systems.

- **NTP (*network time protocol*)**
  NTP is an Internet protocol for synchronising the clocks of computer systems through packet routing, thus avoiding the problems caused by variable network latency. The NTP Project provides NTP support and offers an implementation of reference, available on all GNU/Linux systems.

- **WINS (*Windows Internet Name Service*)**
  WINS allows us to resolve the names of the different Windows services and systems. This function can be replaced by `nmbd`, included in the Samba package.

System management and administration

Most system control and management applications are not native to the operating system and manufacturers often supply versions of these applications for different operating systems. The downside to this is that while there are many systems management applications for GNU/Linux, they are not based on free software.

In all events, the management and control of free software systems is very different to that of systems based on proprietary software, such as Windows. Administrators of free software systems normally use a series of management tools rather than just one, each specialising in a part of the system. Thus, administrators have much more freedom to make adjustments and correct problems with their systems, which is one of the reasons for the well-known reliability and security associated with free software.

An initial option for the automation of administration tasks in GNU/Linux is the use of `cron` and `at`. The former (`cron`) is an administrator of background processes that runs programmes at regular intervals. The `at` command also allows programmes to be run at specific times.

All GNU/Linux systems offer the basic functionality of administration from a remote terminal `ssh` on another client or server in exactly the same way as if it were a local machine, even through the graphical interface of the desktop. The combined use of `ssh`, `cron` and `at` covers many of the maintenance tasks of the administrator.

Other system utilities such as `strace`, `lsof` and `netstat` offer diverse functionalities for detecting and analysing errors, and can be useful in server management.

Network management

The solutions available for the management of TCP/IP networks as free software include Nagios and OpenNMS.

Nagios allows us to monitor servers and services in order to detect network problems in systems based on GNU/Linux. A background process controls the specified services and servers and sends the information to the Nagios server, which informs the system administrator if it detects a problem. By means of a series of plugins, Nagios can actively and passively monitor typical network services such as web and mail servers, besides others such as database management systems.

OpenNMS is a network management application that uses the FCAPS model and allows us to determine the availability of the different services, store the information and generate reports, and inform of events.

However, the management of more complex systems and networks may require the use of tools that are not available as free software.

Software management

Software management involves client installation and restoring, standard and specific applications distributions and the management of updates and patches for the entire system.

The solutions available as free software include `m23`, a software package for systems based on the Debian distribution that allows initial client installation, including the definition of partitions and the detection of hardware, the distribution and updating of software and the restoring of clients.

Web server

Apache is the main alternative for migrating and implementing an organisation's web server. Apache is present in over 60% of web servers and is distributed freely under the Apache licence.

The Apache project

Apache is one of the most successful projects developed by the free software community that, due to certain features of its licence, can be used in proprietary software products.

Its functionalities and performance are excellent and have been thoroughly tested in a wide range of production scenarios. Apache has a modular architecture consisting of a kernel that contains the basic functionalities of the service and numerous easy-to-install modules for specific applications, such as support for certain programming languages (PHP, Java, Perl, etc).

The migration of web projects[29] to an Apache web server requires a study of the individual features of each project, which can sometimes create incompatibilities. Apache offers trouble-free support of both static content (developed in HTML) and dynamic content (developed in languages such as PHP or Perl). The modifications needed to ensure compatibility of these projects in Apache are minimal or non-existent.

(29)The term web project refers to a website (for example, the organisation's website) and to web applications that can be accessed through a browser.

Projects developed in proprietary technologies such as ASP are a special case, since considerable effort is required for them to work on Apache. Wherever possible, it is preferable to implement the web project again in alternative technologies such as PHP to ensure technological independence in the future. This obviously involves more work but the opportunity can be used to optimise the web applications and contents of the organisation.

It is now increasingly common to see PHP used as a web programming language and LAMP platforms (Linux, Apache, MySQL and PHP) have become popular over recent years for offering web contents and applications.

Databases

There are many free software alternatives for the implementation of database management systems, but the most common are MySQL, PostgreSQL, Firebird and MaxDB. Choosing one solution over another will depend on the requirements of migration.

In all events, free databases are mature products that have been tried and tested in production environments and can in fact be regarded as one of the leading areas for free software and the GNU/Linux system in business environments. These solutions also have versions for proprietary operating systems so they could be used when migrating applications only.

Some proprietary databases such as Oracle[30] also have a GNU/Linux version, so in special cases where it is not advisable to migrate the database management system, we could migrate the operating system to GNU/Linux.

(30)Oracle is usually used in fairly complex environments with a series of requirements that cannot always be met by free solutions.

Most databases have fairly standard administration and query mechanisms, which theoretically encourages interoperability and the use of other solutions along with easy migration of data from one management system to another. This means that applications can continue to access the data transparently and without the need for further modifications.

Thus, database migration involves two operations:

- **Migration of data to the new database**. The effort required for this operation will depend on the initial status of the data. If the data can be accessed by SQL queries, an export or data transfer operation with subsequent importing to the new database should suffice. If the data are stored in a proprietary format or even as text files, we will need to implement a parser and subsequently import them to the new database.

- **Verification of data access from applications**. If the applications use a standard data reading mechanism (such as SQL queries), access will have to be set up in the same way, unless we are using commands that do not satisfy the standard. If the applications use standard drivers such as ODBC or JDBC, or a proprietary interface, it will be necessary to replace the original database driver with that of the new database or to implement a new interface. In both cases, this step could require considerable effort and cause problems with interoperability.

As a general rule, for easier database migration, we need to avoid as far as possible the use of predefined query procedures and specific manufacturer extensions for accessing the data from the applications. Instead, it is advisable to use standard drivers like ODBC and JDBC, which are easily interchangeable, and to implement SQL queries in the most modular way possible and separate from the rest of the programme.

Desktop environments and office automation applications

There are two main desktop environment options in GNU/Linux systems: GNOME and KDE.

Both GNOME and KDE provide an intuitive desktop environment with a user-friendly windows manager suitable for all users and a development platform for building applications integrated into the rest of the desktop and between one another.

The choice of one or the other is largely a matter of personal preference. KDE is generally more similar to the Windows interface and has more customisation possibilities, but this could introduce further difficulties for new users.

Although there are many office automation applications for GNU/Linux systems that offer good integration with the Gnome and KDE desktop systems, there are two solutions that stand out above the rest: OpenOffice.org and StarOffice.

OpenOffice.org is an open source, free software, office automation package that is freely distributed. It is available for many free and proprietary platforms, so it is often cited as an example of application migration. In most cases, it is compatible with Microsoft Office and supports the ISO OpenDocument data exchange standard, which can be used freely.

OpenOffice.org is actually based on the StarOffice project. StarOffice is the proprietary office automation package of Sun Microsystems, sold as such, with some additional functionalities[31] not available in OpenOffice.org, which Sun Microsystems continues to support.

(31) Similar *TrueType* fonts to those used by Microsoft, extra templates and picture galleries, additional updates and patches, among others.

OpenOffice.org and StarOffice contain different applications, each with specific function, but they integrate perfectly with one another:

- Word processor (Writer)
- Spreadsheets (Calc)
- Presentations (Impress)
- Mathematic formula editor (Math)
- Drawing (Draw)
- Databases (Database)

OpenOffice.org[32] uses an XML-based compressed file format for all of its applications, which differs from the binary formats used by proprietary office automation applications. With this format, we can easily separate the contents of the file from its data, styles, version control and the pictures included in the document. In addition, OpenOffice.org allows us to work with other formats also based on XML.

(32)Most of the OpenOffice.org features mentioned in this section are also applicable to StarOffice.

Migration of files in Microsoft Office format

OpenOffice.org has mechanisms to convert and import files in proprietary formats, such as those used by the Microsoft Office suite. It also allows us to save files created with OpenOffice.org in proprietary formats.

However, this compatibility is not perfect and while the quality is usually acceptable in most cases, there are sometimes differences in the format of the documents, particularly those containing complex elements, such as macros or other special features. In this case, it may be necessary to re-edit some of the documents if we require the format to be identical to the original.

Hence, before converting and migrating the documents, we need to study their features and classify them according to their use and technical complexity:

- Editable documents: these need to be converted to a new interoperable format such as ODT, so that they can be edited in the future.

- Read-only documents: these can be converted to PDF format, which simplifies the migration process considerably.

- Basic documents: these do not contain macros, proprietary graphics, formats or complex elements or styles such as footnotes, tables and indices. They can be easily migrated by batch processing.

- Complex documents: these contain macros, proprietary graphics and vector graphics, OLE objects, active objects, cross references, etc. They can be migrated but will most likely require individually processing.

OpenOffice.org offers the possibility of converting several documents by batch processing. All of the documents must be located in a source directory and we will need to specify a target directory to which all of the converted documents will be saved. In all events, we recommend checking the conversion with a representative sample taken from all the documents.

In addition, we have two possibilities when it comes to dealing with complex documents:

- Convert the documents one by one so that we can correct any differences with the original document before saving it in the new format.

- Revise the documents one by one to eliminate elements that could affect the conversion process and then batch process them all.

Corporate applications

The term corporate applications is used to refer to applications developed to meet the specific needs of the company or organisation in which migration is being performed.

For migration purposes, we can distinguish between the following types of corporate application:

- Applications that can be run problem-free on a free operating system, such as multi-platform applications (those written in Java, for example) or web-based applications (in PHP for instance, as we saw in the section on web servers).

- Applications requiring slight modifications to be able to run on a free operating system, such as those for correct database access for instance, as we saw in the section on this topic, or for configuring new environment variables.

- Applications that can run by emulation or virtualisation.

- Applications that cannot run on a free operating system, such as applications implemented in languages exclusively for the proprietary operating system.

Most corporate applications are proprietary so the company will not have access to the source code. If it is not possible to run the application by emulation or virtualisation, the best option is to implement the application again as free software based, if possible, on an existing free software project.

2.8. User training, communication and support

Thus far, we have looked mainly at the technical aspects of the implementation of free software systems. The importance of the technology should not detract from the fact that one of the factors in the success of any implementation project, particularly migration, is acceptance of the new system by its users.

This section will first describe the key elements of the free software training plan of an organisation and some good practices to introduce and encourage user acceptance. To conclude, we will take a look at the main channels of communication and key elements of a user support system.

2.8.1. Training

Correct user training plays a very important role in the success of the project. As a result, we should include it in our project plans from the very start.

To plan the training, we need to identify first of all which user groups will use which specific types of application. This will enable us to study the differences between the proprietary and free applications, and hence evaluate the difficulty in the adoption of the new applications for users. These elements will allow us to plan user training to meet their real and individual needs.

Some free applications, such as browsers, e-mail clients and office automation applications, are very similar to their proprietary equivalents. Obviously, less training will be needed in these cases.

Materials

There are many materials on the Internet that can be used to train users or to help us prepare our own materials.

Manuals and documentation are often only available in English, which can be a problem for some users, and the interfaces of certain applications are not translated or the translation is incomplete. In this case, we could think about publishing our own documentation to overcome language barriers.

The European SELF project has a platform for creating and sharing educational materials relating to free software and open standards.

Training manager

The training may be held in the organisation, in collaboration with an outside company or using an online learning platform.

In any case, it is important to facilitate access to training and materials as much as possible. Attendance of training activities may or may not be compulsory as this will depend on the organisation's policy.

Online learning platforms have the advantage that users can adapt their learning process and training access to their needs. Moodle is a free course management system used to create what are known as **online learning communities** in which students can follow the training and communicate with one another.

Another good option is to combine on-site training with an online learning system.

Finally, we should not rule out the possibility of offering users some sort of incentive to encourage them to attend the training, for example, awarding certificates of attendance and achievement.

Types of user

Not all users are alike. Firstly, there will always be some users who are more receptive to the new software than others. In most cases, however, once users overcome their initial misgivings about the use of free software, they find it very similar to proprietary software and are satisfied with its use, so we should not worry too much if their first experiences are negative.

Secondly, remember that technical and non-technical staff will need a different type of training and monitoring.

Technical staff will need more intensive training than normal users, particularly if they have no previous experience of free software and are used to working with a proprietary system that, in contrast, they are very knowledgeable of. The participation of technical staff is also key to ensuring the smooth running of the system after implementation. It is good practice to motivate them and get them involved in the implementation process so that they can get to know the system while it is being implemented.

2.8.2. Introduction to free software

In addition to training, another of the practices that can lead to a successful free software migration project is the gradual introduction of the new applications and services. This gives users time to get used to the new environment and means that they are not confronted by a completely unfamiliar system.

Installation of bridge applications

Many popular desktop applications are now available on GNU/Linux as well as proprietary systems, such as the OpenOffice.org office automation package, the Firefox browser or the Thunderbird e-mail client. Many server applications and services can also be run on both systems, such as the MySQL database manager or the Apache web server.

These types of application are called *bridge applications* and can be very useful in the early stages of migration[33] and for evaluating the response of users and estimating their training needs much more accurately.

(33) This would be a simple applications migration, like the one we saw in the section on types of migration.

Staggered migration of services

The primary aim of any migration is to produce a smooth transition from one system to another without the users noticing any major differences or, if possible, without them noticing any differences at all. One strategy for achieving this aim is to start migration on the servers, so that users can continue working as normal until the system is prepared for the migration of clients.

Services that can easily be migrated at the start include network services (DNS, DHCP, etc.), web servers and database servers. It may be necessary to use technological solutions that work well on heterogeneous systems, such as OpenLDAP in combination with Samba.

This gives us enough time to train technical staff too, whose support can be very useful for migrating clients and providing support to other users.

Both the introduction of bridge applications and the staggered migration of services should be taken into account in the project planning.

2.8.3. Project communication

As we have seen, the implementation of a free software system and migration to this system is a process that involves all of the organisation's users, not only the technical staff responsible for its complete introduction and maintenance.

The mechanisms must be put in place for effective communication between users and the technical and administrative managers of the organisation, and we must ensure transparency throughout the process. As of result, the communication activities must be set down in the project plan, which must include:

- **Initial group communication to all users**. We should use informative meetings, notes, internal e-mails and advertisements on the organisation's intranet to explain the reasons for the project and detail its general planning before it is implemented.

- **Regular communication of the project's progress**. We should explain which parts of the system will be migrated and when, along with any changes made to the project. Meetings with a small number of users should be organised at each phase of migration.

- **Regular meetings after conclusion of the project**. We should evaluate the success of the project and conduct general monitoring of its results and the experiences of system users.

2.8.4. User support system

A key element of the new system is the introduction of an issue management system for users, allowing them to find the answers to their questions and solve the technical problems caused by the situation. It is important to give a fast and efficient response to all these problems, particularly just after implementation.

When designing a user support system, we need to answer the following questions as they will define the main features of the system:

- Who are the users?
- How does the organisation operate?
- What type of support do the users need?
- What type of support is offered to each type of user?
- How much support will be offered?
- How will the support be offered?

Pilot tests can be used as a basis for characterising most of the problems encountered by users and for preparing a procedure for solving each one.

Similarly, we will need to identify the critical services and users of the system, as these will be given preferential support.

We also need to bear in mind that more support means higher costs. It is possible to provide more support in the weeks immediately after migration, when the number of queries and issues is higher. In all events, the key to an efficient user support system is fluid communication with users, to make them aware that their problem is being looked into.

Lastly, it may be the case that a user support system was in place before migration existed. If this system was based on proprietary software, we will need to evaluate the diverse free alternatives.

Website

There are many solutions. For a comparison of some, see http://en.wikipedia.org/wiki/Comparison_of_ticket-tracking_systems.

3. Free software companies

This third unit of the "Implementation of free software systems" subject outlines the main concepts and characteristics associated with the business side of free software.

Ever since it emerged, free software has always been present in information technologies and its evolution has been influenced by the structural changes that have taken place in technology, economics and society.

See also

To find out more about the history of free software, see point two of the materials for the "Introduction to free software" subject.

As time has gone by, a number of philosophies on the creation, production and diffusion of software have been developed. We can generally distinguish between two basic and antagonistic trends:

- On the one hand, the proprietary philosophy, which defends the protection of software by closing and privatising the source code, combined with the use of licences with heavy usage restrictions.

- And on the other, the free philosophy, which defends the freedom of the software and source code with licences guaranteeing user rights to run the programme, study and adapt the source code and redistribute and publish any improvements made to it.

These two philosophies have generated business models with conflicting ideologies, operation, development and economics:

- The privative software model normally establishes a financial value that has to be met by restricting the use of a binary format copy, which makes it impossible for people or organisations without copyright or the explicit authorisation of the copyright owners to adapt, correct or improve the source code. Many proprietary licences prohibit the transfer of user rights to third parties without the prior agreement of the copyright owners.

- The free software model tends to focus on the development and adaptation of free and qualitative solutions to meet the needs of users and organisations, and on complementary services for their implementation and day-to-day operation. Hence, the business model based on free software allows actions that are prohibited or restricted in the privative software model.

The unique philosophical understanding of free software does not only have a direct impact on the business model and business strategy, but also on the definition, management, organisation and operation of technology companies. Aspects such as the maturity of free software, the presence of a

world community of free software project collaborators and the viability of its business models seriously question the traditional concept of a business project.

The first section of this unit defines the different business models based on free software, which are valid and viable to be put into practice as a business strategy.

The second section focuses entirely on the drafting of the business project and details the main aspects of the creation, organisation, production and operation of a free software company.

The third section introduces free software production and describes the main features of the creation, organisation, communication and development of the source code.

Lastly, this unit has two appendices that briefly and systematically describe the main free licences and open standards directly related to the free software business.

3.1. Business models

This first section introduces the main free software business models, together with the characteristics and features setting them apart from business lines based on proprietary software.

By and large, the biggest difference between free and privative software from a business perspective is the licence. Broadly speaking, a licence is a contractual model by which the author of the product (or whoever owns the copyright) establishes the rights and duties of the users of the product and the scenario in which it can be used.

However, free licences[34] are based on four basic principles of freedom that relate to the software and its source code:

- The freedom to run the program for any purpose.
- The freedom to study the source code and adapt it to one's own needs. Hence, access to the source code is necessary.
- The freedom to redistribute copies of the software.
- The freedom to improve the software and release improvements. Hence, access to the source code is necessary.

(34)There is a debate between the Free Software Foundation (http://www.fsf.org) and the Open Source Initiative (http://www.opensource.org) over the implications of the terms *free* and *open*.

Internet resource

You will find the original definition of free software at http://www.gnu.org/philosophy/free-sw.html

The basic freedoms of free software conflict with the privative model focusing on the sales of licences for restricted use of the binary format[35], although free software does not necessarily have to be obtained without payment. However, much of the free software currently available can be obtained by direct, free download from the Internet site of the organisation that manages it.

(35)This is known as right-to-use licensing.

Direct or free downloads

Examples of direct or free downloads include:

- Debian GNU/Linux from http://www.debian.org/distrib/
- FreeBSD from http://www.freebsd.org/where.html
- KOffice from http://www.koffice.org/download/
- OpenOffice.org from http://download.openoffice.org/

The philosophical opening up of free software encourages business models based on human capital, knowledge, customisation and the adaptation of products, not to mention the constant evolution of the software. Hence, we need to highlight the role played by the community of free software users, which helps monitor the quality and evolution of free applications with a level of performance that would be difficult to match in proprietary models.

Over time, the free software model has managed to consolidate an offer that covers most sectors with a privative software presence, shaping a mature, qualitative and secure market on which to base a business strategy covering both software development and complementary services.

Internet resources

At http://freshmeat.net/ and http://sourceforge.net/, you will discover a wide range of free software projects in the key areas where the technology is used.

To an extent, the free software business strategy bases itself on the aspects that set it apart from the proprietary model, such as increased functionalities, tailored adaptation, numerous and continuous revisions, guaranteed product security and quality of operation, and a whole range of complementary services for its implementation and day-to-day operation.

The following sections introduce the main business models deriving from the conceptual philosophy of free software: development, consulting, installation and integration, migration, maintenance, and support and training.

The business models described should be seen as complementing one another rather than being independent. In other words, a combination of one or more business models may be necessary to cover the business strategy.

3.1.1. Development

The software development business model involves the total or partial production of a product based on free software in order to be marketed either directly or as part of a third-party implementation project, such as those described in the second unit of this subject.

The definition of free software makes no reference to the strategy of selling a free product at a price per copy sold but the characteristics of free licences make this a secondary option, albeit one used intensively by some organisations.

Packaged free products

Some organisations decide to offer their free products in packages (boxes, discs, manuals, documentation, etc.) in exchange for the payment of an amount that, while lower than the price of alternative solutions, is still greater than cost price. For example, the Ubuntu distribution can be purchased at http://www.ubuntu.com/getubuntu/purchase.

Free software production is mainly a response to the sale of complementary services with added value for clients, which also serve to extend the continuity of the software, such as customisation or adaptation to a specific environment.

The materials for the "Introduction to free software" subject include a classification of the possible alternatives to the development of free software, which we will summarise below:

See also

You will find more information and examples of this classification in Section 5.2 "Business models based on free software" in the materials of the "Introduction to free software" subject.

- **Better knowledge**. This is based on the idea of doing business with the knowledge of one or more free products, offering tailored developments, modifications or adaptations (among others that we will describe later). Active participation in the creation and development of free products is the added value introduced by the company in the eyes of its clients and the competition.

- **Better knowledge with limitations**. This is similar to the previous model (better knowledge) but with a mixed implementation of free and proprietary licences (or patents) to reduce competition. This model may prove not to be viable if the free product forks into a branch supported by the free community, causing the competitive edge to fade.

- **Source of a free product**. This is similar to the first model (better knowledge) but with the difference that the company produces almost all of the code. The client evaluates the positioning and competitive advantage over the competition. This model receives the support of the free community.

- **Source of a product with limitations**. This is based on the previous model (source of a free product) but with an implementation aimed at

reducing competition, such as beginning distribution with a privative licence and opening it up afterwards or limiting the initial distribution to company clients.

- **Special licences**. This involves the production of a single product distributed under different licences (free and proprietary). The proprietary product offers special implementations of the product, such as integration with other proprietary products.

- **Sale of a brand**. This involves the distribution of free products with the image of a corporate brand, offering quality and added value from the point of view of clients. These products are usually accompanied by numerous complementary services for clients.

The choice of the type of software development business must go hand in hand with the business strategy and be suited to the target market. Hence, an organisation may decide to use a customised typology for each of the products it plans to introduce on the market, based on the strategy and specific target market of each product.

The free software development business model also requires a careful selection of the licences of the source code it uses:

- The licence for the source code it modifies if the end product improves an existing application.

- The licence for the source code it links to if the end product needs to implement function calls to external libraries in order to operate.

- The licence for the source code it creates, i.e. when the source code is completely new.

- The licence for the source code of the end product, which encompasses the combined source codes of the end product.

The importance of carefully determining the licences associated with each part of the source code used lies in the differences between the various free licences. In other words, although all free licences guarantee the basic four freedoms, they differ in their policies for licensing the redistribution of modified code, which is precisely the object of the free software development business model.

Appendix I of this unit briefly describes the most important characteristics of the main free software licences and explains the redistribution policy and the compatibilities of linking and derived work.

The selection and correct combination of licences will have a direct effect on free software production and may have legal implications if it is not done properly. The last section of this unit on free software production looks in detail at how to select the relevant licence based on the product parameters.

Lastly, we should be aware that free software promotes and uses public specifications called open standards to promote the universality and interoperability of the formats handled. Appendix II of this unit includes a brief description of the main characteristics of open standards together with some examples.

3.1.2. Consulting

The consulting business model is based on the generation of professional services to complement free software for users and organisations.

To a certain extent, this business model is based on providing quality external professional technology services to organisations that are not fully in control of the creation, management, development and evaluation of their internal technology projects.

See also

For more information on the typology of production projects, see the "Classification by scope" section of the first unit.

Consultancies can offer a broad range of professional services, which will depend on the strategy and context of the business. Nonetheless, they will be closely related to the study, analysis, design and evaluation of the free software systems implementation project described in the second unit of this module.

There now follows a brief classification of the main services that consultancies can offer their clients:

- **Project management**: this involves the creation and functional management of the free software implementation project. The tasks carried out as part of this service cover the life cycle of the project, management of the teams of professionals involved at each stage of the project, control of the effective progress of the project and generally all tasks concerning the coordination, information, management and monitoring of the project.

More information

The "Project typology" section of the first unit and the "Life cycle" section of the second unit contain more information on the management and life cycle of free software projects.

- **Execution of the project analysis and design stages**: this requires carrying out one or more analysis and design stages of the free software implementation project. The tasks of this service are those that the organisation outsources to the consultancy, such as the study of the current system, study of the requirements of the new system, analysis of

free software solutions and/or design of the new system, in accordance with the stages indicated in the second unit of the module.

See also

The "Study of the current situation", "Study of the implementation requirements", "Analysis of free software solutions" and "Development" sections of the second unit contain more information on the study of the current system, the study of system requirements, the analysis of free software solutions and system design, respectively.

- **Evaluation and auditing**: this requires carrying out professional technological assessments of one or more characteristics of systems in operation. The tasks of this service may be typical of a systems implementation project, such as the execution of stages we saw earlier, but they can also be carried out independently and in isolation. The evaluation or audit evaluates one or more aspects of the system, such as security, performance, efficiency or efficacy, among others, and may be performed before and/or after implementation of the system.

- **Advice**: this service is geared towards offering support and professional help and advice for technological decision-making in the organisation. These tasks can be carried out prior to the start of any implementation project or during the study, analysis and design phases. In all events, they form part of the professional support provided for strategic decision-making on technological aspects affecting the future of the organisation.

This list is by no means exhaustive or exclusive because the business model can provide two or more services to cover its business strategy. In addition, the consulting business model can be combined with other business models to offer clients a comprehensive technology service.

Hence, the best knowledge of the technology platforms in place (or to be implemented), and excellence in the analysis and extraction of information and conclusions, or the scope and complexity of the project, are decisive characteristics that affect an organisation's decision to outsource management or stages of implementation.

Consulting work is normally formalised through open or closed contracts. In open contracts, the relationship begins when a specific service is commissioned and, depending on the result of this service, the contract may be extended with the commissioning of new services. For example, open contracts may be used for the execution of one or more stages of the project.

In contrast, closed contracts are awarded for the completion of a specific aim, task, result or assignment and there is no direct possibility of extending the contract in the same scenario. For instance, closed contracts may be used for the independent auditing of a system, given that these tasks are carried out in isolation and at specific points in time.

3.1.3. Installation and integration

The installation and integration business model is based on the direct implementation of free software systems for users and organisations, usually as part of free software projects.

In a way, this business model considers free software as the object of the production of its services, rather than a product in itself. This creates a market offering substantial benefits to clients:

- The organisation does not have to pay for licences for free software products that are freely distributed so it can cut the costs of technological implementation.

- The organisation does not need to practice product piracy and thus will not breach the applicable legislation.

- The organisation can directly adapt the free solutions, thus cutting the cost of the implementation of specialist systems.

- The organisation can adopt integrated direct implementation packages, and hence reduce the risk of technological implementation.

A number of services can be offered under this business model, the main ones of which are listed below, although this list is not exhaustive and does not exclude other services:

- **Configuration**: this model carries out the tasks of setting up and fine tuning[36] a system already in place in order to formalise the initial set-up, enhancing its performance or adapting it to new purposes not considered initially. In all events, the fine-tuning does not affect the source code of the application, only the configuration of the components that can be adapted to the specific features of the installation.

(36) Also known as *set-up* or *tune-up*.

- **Tests**: the aim of these is to provide benchmarking for systems, applications or free software solutions from a given perspective. We may need to conduct a comparative analysis of free solutions, testing of a new system design or testing of directly implemented software on specific hardware, either as part of an implementation project or as independent, one-off tests.

- **Integration**: this involves carrying out and/or checking integration between two or more free software solutions in order to provide a single package to resolve a specific operating function[37]. This integration can usually be resolved with a configuration to suit each element and possibly an additional component allowing more efficient integration.

- **Installation**: this involves the bulk installation of software for direct implementation on client machines or servers[38]. This service may require the configuration and fine-tuning of both the free software to be installed and the hardware on which it will be installed. It may also be necessary to integrate the diverse solutions we wish to install. When the same software needs to be implemented on a series of computers with identical hardware, it may be useful to use pre-configured image distribution and cloning software to save time and money and to improve the efficiency and efficacy of the process.

- **Distribution**: this involves redistributing free software to clients, either in the original format[39] or in customised configurations related to the scope of the business, such as tool integration, operation geared towards clients, servers or a work station, among others. The redistribution of integrated software is subject to the licences of the specific solutions. Appendix I describes the main free software licences and their compatibilities.

More information

The "Analysis of free software solutions", "Development" and "Implementation and migration" sections of the second unit contain more information on the analysis of free software solutions, system design and system implementation and migration, respectively.

(37) For example, LAMP (*Linux, Apache, MySQL, PHP*) is an integrated software package with diverse individual aims but which, as a whole, solves a specific problem efficiently and effectively.

(38) This is similar to the term *Installfest*, though applied here to a structured business.

(39) At http://freshmeat.net/ and http://sourceforge.net/, you will discover a wide range of free software projects in the key areas where the technology is used.

As we can see from the above classification, these services are closely related to the stages of free software solutions analysis and of implementation and migration of the implementation project described in the second unit. Hence, a better knowledge of the technology platforms and excellence and efficiency of services or the scope and complexity of the project are decisive characteristics affecting an organisation's decision to outsource the installation and integration of its system.

3.1.4. Systems migration

The systems migration business model is based on transferring the operating function from the system in place to the system to be implemented.

Migration is a complex process that must be carried out with accuracy and rigour, since the data and configurations we are dealing with are the organisation's capital.

The diversity of situations encountered by companies that migrate systems are the result of a combination of the source and target platforms of migration. A thorough knowledge of the platforms and experience in migration are the basis for offering added value to clients.

The following list indicates the main services offered under this business model, although it is not exhaustive and does not exclude other services:

- File system services, both for the server and clients.
- Printing services, between clients and between servers.
- Directory services and centralised authentication services.
- Network services, particularly automated management protocols for network control, communications and clients.
- System management and administration, for network and software management.
- Web services, for static and dynamic platforms.
- Databases, for data migration and access verification.
- Desktop environments and office automation applications, for applications and user data.
- Corporate applications, for applications that can be run directly and those that require tuning or virtualisation.

See also

The "Implementation and migration" section of the second unit details the characteristics of systems migration. In the "Migration of the services of a system" section, you will find more information about the services described here.

The complexity and scope of migration, the ability to carry out the process carefully, efficiently, effectively and as quickly as possible, and the best knowledge of the source and target technology platforms of the migration are decisive aspects that can persuade organisations to outsource migration of their systems.

As we can see from this classification of services, the migration process may need other services, such as installation, configuration, integration or testing to ensure that we meet our aims.

Free software uses and promotes open standards for interoperable data exchange and its role in systems migration is particularly important. For instance, starting off with a system that does not store data in open standards could complicate migration because of format conversion, especially if the original is privative. Appendix II of this module introduces the open standards, defining them and the organisations behind them, and offers some examples.

3.1.5. Systems administration and maintenance

The systems administration and maintenance business model involves carrying out management and monitoring tasks for a system already implemented and operating.

The main aim of the services offered under this business model is to keep the entire system up and running, adapting the configuration to changes, solving any problems that may arise and repairing malfunctions that affect the normal operation of the organisation.

The following list indicates the main services offered under this business model, although it is not exhaustive and does not exclude other services:

- **Administration**: consists of providing basic management of the system, the adjustments required as time goes by, supervision of its operation, the implementation of new functionalities and the evolution of the system. Many administrative system tasks can be carried out remotely[40].

(40) For instance, through the combined use of `ssh`, `cron` and `at`.

- **Maintenance and evolution**: consists of supervising, monitoring and redressing issues in the system that could affect its operation, together with the control and evolution of the obsolescence of its components. Examples include malfunctions and the deconfiguration of hardware or software, the control and updating of software versions, and the evolution plan for the hardware and software.

- **Security**: this consists of managing the security of the system, controlling risks, maintaining policies for prevention, contingencies, diagnosis and debugging. Examples include backups or the control and maintenance of keys and certificates.

Given the characteristics and features of these services, many organisations decide to keep staff on for these tasks, but some small and medium-sized organisations are unable to create such a position.

The outsourcing of some services, such as intranet and extranet servers, can also lead to the contracting of external administration and maintenance services. These services are normally contracted for a fixed monthly or yearly fee and cover a specific service level.

Intranet and extranet

Intranet and extranet web services are easily outsourced because of the proliferation of *data hotels* and *data centres*.

3.1.6. Support and training

The support and training business model involves providing professional technical assistance for the technological training of users and the resolution of issues and problems relating to use of the system.

The implementation of free software systems may initially require user support and training measures, particularly if the previous system was based on privative software. As we saw in the second module of this subject, the implementation project must take into account the need for user training in order to promote positive change management, whose features make it a service that can be easily outsourced to companies specialising in this sector.

The following list indicates the main services offered under this business model, although it is not exhaustive and does not exclude other services:

- **Training**: this service provides education and training on free software tools, including operating systems and office automation tools, to end users. The service can also include specialist software training as a result of the development of the systems implementation project, so it may be useful for change management purposes to coordinate this task with the implementation team.

- **Support**: this service provides technical assistance to users in order to solve everyday problems. Many of these services are provided from call centres, but it can be a good idea to provide e-mail addresses for resolving issues or instant messaging with professionals. It can also be convenient to combine these tasks with those of the systems implementation project in order to fix possible bugs in the implemented software.

See also

The "User training, communication and support" section of the second unit contains more information about user training and support.

This business model generally requires human, technological and material resources adapted to the aims of the training:

- Human resources with an deep knowledge of the issues and the ability to transfer knowledge and solve problems.

- Technological resources suited to training and support, such as technological platforms for learning or call centres offering technical assistance.

- Material resources adapted to the training, such as specific documentation and manuals on free software[41] with free licences.

Internet resource

Moddle is an example of a virtual learning platform based on free software. http://moodle.org/.

(41)The European SELF project has a platform for creating and sharing educational materials relating to free software and open standards (http://selfproject.eu/).

The quality of these parameters is essential if the organisation is to outsource its training and support services. Training services are usually contracted in the form of courses whose structure is agreed previously, while support services are contracted for a monthly or yearly fee, following agreement on the services covered.

See also

Section 10.2 "Licences of other free resources" of the "Introduction to free software" subject material contains two licences for documentation, materials and literary works that are widely used.

3.2. Business plan

A business plan or project is an instrument that identifies, describes and analyses a business opportunity, studies its viability and develops the procedures and strategies for creating the company to exploit this business opportunity.

Clarification

In this section, we will use the term business plan because our aim is to describe the elements required to set up a free software company, as in the case of Cometa Technologies, which we shall see in the second section.

Taking into account this definition, the aims of the business plan will be as follows:

- To conduct a market study to position the business plan and determine its technical, economic and financial viability.
- To develop the measures required to achieve the aims set out in the business plan.
- To monitor the evolution of the company and analyse deviations from the initial business plan.
- To serve as a calling card for the project and the business entrepreneurs in order to obtain the financing and support of third parties.

Although the first three aims are mainly internal, the last is external and visible by people who do not form part of the project, in theory at least. When drafting our business plan, we always need to consider this dual aim: to act both as a plan for the project and as the presentation of the project.

Naturally, we need to avoid falling into the trap of omitting the risks or negative parts of the project to make it look more attractive to investors. In fact, missing out these elements could be detrimental to our business project because it would be based on false suppositions. A true picture of the technical and economic aspects is thus one of the basic requirements for drawing up a business plan.

Any business plan needs to respond to a series of questions about the project we wish to introduce: Who?, What?, Why?, Where?, When? and How much?

- **Who?**

The name of the company, the brand of the products or services offered, the names and track record of the business developers.

- **What?**
  The description of the products or services offered, the markets at which they are aimed and the market share set as the target, among others.

- **Why?**
  Every business plan generally seeks to obtain and maximise profits. However, this is not incompatible with other aims, such as improved quality of life in society or the creation of jobs.

- **Where?**
  The geographical area where the products or services are to be marketed, which can be regional, national or international. The distribution channels that will be used, including possible agreements with other companies who will allow entry to other regions.

- **When?**
  The expected start of business and subsequent planning, including temporary conditions or limitations that could affect the company, such as procedures for obtaining licences, production time, obsolescence of certain technologies or seasonality.

- **How much?**
  The initial investment needed to launch the business project, the minimum and desired turnover, the threshold of profit and loss, the reinvestment of profits and the sharing of dividends, among others.

These issues are covered in the following aspects, which are found in almost any business plan:

- Executive summary
- Introduction
- Business description
- Organisation of production
- Internal organisation and human resources
- Market study
- Marketing plan
- Financial analysis
- Legal form
- Risk management
- Summary and evaluation

Depending on the nature of the company or business, these aspects will have a greater or lesser importance in the business plan and may be organised in different ways.

The following sections will discuss each of these aspects and study their relationship with the free software business models we saw in the previous section.

3.2.1. Executive summary

An executive summary is a short statement[42] that appears at the start of the business plan and summarises the main points of the document. It gives potential investors a comprehensive idea of the business plan without having to go through the various sections in detail.

(42) In all events, the executive summary should not be more than three pages long.

The executive summary should cover almost all of the points of the business plan, which are:

- Description of the business model, with a particular emphasis on the chain of value and source of income.
- Short description of the project developers, their training, knowledge and skills, professional track record and dedication to the new project.
- Concise description of the market, including size, clients, growth potential and barriers.
- Analysis of the functional areas of the project: production, quality and organisation of human resources.
- Summary of the financial analysis of the project and the investment required to set it up.
- Summary of the risks of the project and the plans to prevent them and remedy their consequences.

Obviously, the executive summary should highlight the strong points of the business plan, particularly for the business model we wish to adopt, the strategy that we will use to do so and the team developing the idea.

We recommend writing this part after completing the business plan and to do it from scratch, that is, without re-working texts that we have already written.

3.2.2. Introduction

After the executive summary and the index, the first part of the business plan should be an introduction indicating the name of the future company[43] and the team of developers, together with the other professionals involved in the drafting of the business plan.

(43) If the plan describes a new project or service for an existing company, it is a good idea to include a summary of the company's activity, history. evolution, size, etc.

As we have seen, the presentation of the team of developers should cover the professional career of each of its members and the knowledge they will bring to the business project. More often than not, the team of developers will have members with a profile specialising in business management and others specialising in specific technological areas, as is the case of companies that work with free software.

Lastly, the introduction should provide a brief description of the different sections of the business plan that will be developed later.

Mission and vision

The introduction is the place to describe the mission and vision of the new company because it allows the reader to see how these two concepts are developed in the business plan.

The mission and vision of an organisation are a concise definition of its main features and aims, and its strategies for meeting the latter.

The mission is a short phrase justifying the existence of the organisation, i.e. the basic aim of its activities and the values guiding the activity of its employees. The mission is closely linked to the internal values of the organisation and basically describes how to compete and generate value for customers.

The vision is also a concise phrase describing the medium- and long-term goals of the organisation. It is addressed to the market and should offer an expressive and visionary angle on how the organisation wishes to be seen by the world.

The main differences between the mission and vision can be summarised as follows:

- The mission describes the internal aspects of the organisation and its operation, while the vision describes the external aspects.

- The mission has a short- to long-term time horizon and highlights the aspects that should be put into practice immediately in the organisation,

while the vision is fixed in the medium- to long-term and gives the general lines of the future evolution of the organisation.

Corcaribe Tecnología and eZ Systems

The company Corcaribe Tecnología specialises in products and services based on free software and has the following mission:

"Corcaribe Tecnología provides technological solutions that generate added value with a business model that allows us to offer the best cost for results delivered to our clients, producing authentic tangible and intangible benefits for our members and collaborators."

And the following vision:

"To become a Latin-American reference of continued success in the implementation of integral technological solutions, applying the principles and values of free knowledge in a model of sustainable development."

Similarly, eZ Systems, which provides free content management software sets itself the following mission:

"To be the leading content management platform by 2012."

And the following vision:

"To help companies to manage, publish and share information."

It is not essential to define a mission and vision in the business plan, but it can help to summarise the short-, medium- and long-term aims of the business project and to convey them effectively to potential investors.

3.2.3. Description of the business

We recommend you should begin this section with a description of the company you wish to set up and a brief presentation of the project developers, even if you have already done so in the introduction.

The main aim of this section is to describe the products or services for which you are drawing up the business plan and the business model adopted to offer them, as we saw in the previous section.

Special care should be taken when explaining the features of business models based on free software since the reader of the business plan will not necessarily be familiar with them. This includes aspects of the protection of intellectual property and rights over products and services.

It is also a good idea to describe the needs that will be met by the products or services and point out the main differences with the existing offer, in order to show that the business project is well positioned in the market.

Lastly, we will need to indicate the capacity for the production and provision of services, which will serve as an introduction to the next section on the organisation of production.

3.2.4. Organisation of production

The section of the business plan on the organisation of production describes the technical tasks of the future company.

This far, we have looked at the business plan from the point of view of describing the marketing of new products and services, without distinguishing between them. Now, however, this section of the business plan will take one of the following forms, depending on whether it concerns products or services:

- If the company business is the development, production and subsequent marketing of a product, we need to detail the development and production phases.

- If the company provides a service that does not involve a production process, the procedures for provision of the service and the technical needs should be described in detail.

It goes without saying that the two options are not mutually exclusive and a business plan can contain both. For example, a company might specialise in the migration of systems to free software but also provide training in free software technologies to users and technical staff.

Generally speaking, a business encompassing the phases of research, development and production will be much more complex and involve greater risks:

- **Research and development phase**. When describing the research and development phase, we need to pay special attention to the estimated duration of the research and development phase and the necessary investment in human and material resources.
  In high technology sectors in particular, as is the case of free software, the business plan must evaluate the human resources skills and know-how needed for the successful completion of research and development tasks. This section should also detail the distribution of roles and duties, the risks inherent to all research and development activities, the potential synergies between projects, the process of innovation and continuous product improvement and how this process will be integrated into the production process.

- **Production phase**. The description of the production process should deal first of all with the operating cycle[44], the location of the production facilities, their cost and their accessibility. Secondly, we will need to describe the business premises, buildings and equipment needed for the production or provision of services. For each of these, we will need to indicate the modes of financing and purchasing[45], their features, availability, useful life and annual amortisation.

(44) This includes production capacity in number of units and expected production, along with the staff and number of hours or shifts needed for production.

(45) We can also present expansion plans for the facilities and the purchase of new equipment.

Special attention must be paid to quality management. Here, we should describe the quality standards and certifications that will be applied to both the processes and the results of the production process.

Lastly, we need to offer a strategic vision of the production process; for instance, if we are outsourcing production of certain components or part of the production process[46].

(46) For example, a free software publisher could outsource production of the distribution medium and packaging of its programs.

Again, the description of the free software production process reveals a number of differences with proprietary software development, which should be explained in detail in the business plan, particularly when the reader is not familiar with free software. We can also stress the added quality of free software when compared to proprietary software.

More information

The "Free software production" section of the third unit looks in detail at the special features of the production process for free software.

In all events, remember that you should always explain the advantages and disadvantages of the various alternatives and justify each of your decisions.

3.2.5. Internal organisation and human resources

This section of the business plan details the organisation of the team needed to develop the business project and the profiles required for it.

Firstly, we need to include a description of the key duties and management positions, along with the necessary profiles and even the name and professional background[47] of the people who will fill these positions if they have already been selected. We must then describe the categories of professionals required by the company, their duties, main tasks and the type of contract they will have. It is a good idea to indicate the salaries of each type of worker, regardless of whether or not they are management positions.

(47) This includes their professional experience, specialisation and main professional achievements. The purpose of this type of information is two-fold: on the one hand, it boosts the confidence of potential investors and, on the other, it allows us to pinpoint the strengths and weaknesses of the management team.

The internal structure of the company can easily be represented on an organisation chart by departments and business areas, naming the individuals occupying the management positions, if these have already been filled.

This section should conclude with a description of the company's general human resources policy and indicate whether the creation of a specific human resources department is necessary or, alternatively, this function can be split across the different departments.

The need for and availability of qualified staff in a given area and at an acceptable cost can sometimes be a major obstacle, which is the case of specialists in free software[48].

(48)For example, when it comes to publishing these materials, although free software is well known and technologies and solutions based on it are fairly popular, it is difficult to find professionals who have actively participated in free software projects, whether as employees of a company or on their own initiative.

Furthermore, a company that bases its business model on free software may require the creation of positions and responsibilities that suit its specific features. For example, besides the traditional positions of technical director or sales director, we can come across roles such as community director, the person who manages relations with software developers and users, or director of cooperation projects, who manages and coordinates projects developed in collaboration with other companies, research centres or universities.

3.2.6. Market study

Market studies are an essential part of business plans and hence one of the keys to its success.

A good market study will help us to assess the technical and financial feasibility of the business project correctly and to identify potential clients and competitors in order to come up with the right strategy for marketing the products or services detailed in the business plan.

When drafting a business plan, it is useful to conduct the market study first, at least as a rough version, because its results can affect various parts of the business plan.

Thus, the market analysis needs to provide information on the following aspects:

- **Current market situation**. We must first segment the market according to the most relevant features of the business plan and determine its size together with its past evolution. We will also need to determine the decision-making process of the market and the behaviour of its customers, particularly their reaction to the launch of new products or services.
  Secondly, we need to evaluate the needs that may arise as a result of the introduction of the products or services proposed in the business plan. This depends largely on whether the product or service offers something new and on our ability to influence customer habits.

- **Market growth forecasts**. Once we have established the current status of the market, we need to be able to predict its future evolution. Is this

market developing, stable or declining? How fragmented is the market? Is the market becoming concentrated?

Again, we need to take into account the potential influence of the new products or services on the market. For example, the introduction of solutions based on free software can effectively change the market because they encourage the formation of a new sector specialised in free software.

- **Identification and classification of clients**. One of the primary aims of market analysis is to discover who the potential clients of the proposed products or services are. The task of classifying the diverse types of client based on common features is also very important because it allows us to define different strategies for each. For example, a company that implements free software systems in companies will present itself differently to clients depending on whether it is a family business or a major corporation. We also need to remember that a product or service may be offered to clients who are theoretically different. The flexibility and interoperability of free software encourages this type of action.

  Moreover, we need to evaluate each type of client's reception of the product or service. Continuing with the previous example of a company that specialises in the implementation of free software systems, a major company with dedicated technical staff may be more reluctant to use free software, partly because of the fear of change, whereas a family business might be more receptive.

  Lastly, if the future company already has a client portfolio or has clients who have expressed an interest in its products or services, this can be included in the market study.

- **Analysis of the competition and its products**. A market study should reveal the competitors of the future company and identify their strengths and weaknesses as well as those of the products or services they offer.

  We must indicate the characteristics of their products and services, including price and quality, and their market share and sales strategy. It is very important to identify the market leaders for each of the products or services covered in the business plan.

  We should also be careful not to disregard potential competitors, i.e. companies that are not yet on the market but which could enter, or companies from other geographical regions. In our current climate, particularly in information and communication technology sectors, as is the case of free software, the competition tends to be global and many companies can offer their services directly or indirectly from any location.

- **Analysis of barriers to market entry**. Barriers to market entry are obstacles that companies come up against when they enter a new market, such as the need for major investments in the case of new companies or the lack of an established brand. An example of this is that free software tends to suffer from a perceived lack of quality in contrast to the proprietary software companies and solutions already on the market.

In the same way, however, we can also study which barriers to entry we should focus on once we are on the market in order to keep the competition at bay.

- **Influence of governments**. The market study should discuss the way in which local, regional, national and international governments can influence the market and, hence, the viability of the business plan. For example, governments can act both as market regulators and as providers and clients.

  This is particularly true in the case of free software, which, as we have seen throughout this subject, is a point of interest for many governments, including the regional government of Extremadura and the Federal Government of Brazil.

  Market studies need to be planned carefully because they involve a number of phases, which can be summarised as follows:
  - Collection of general information, by which we obtain a large volume of data on the market under study.
  - Analysis of the information obtained.
  - Selective search for information to obtain the missing information required to complete the market study, which will have been identified in the previous analysis.

  To conduct the market study, we will need a great deal of information and this is not always easy to obtain. There are countless bodies and sources of information, both general and specialised in specific regions and sectors: governments and national statistics institutes, local and regional governments, private bodies such as chambers of commerce and business associations, journals and specialist publications.

  A good market study should conclude with a strategic analysis[49] that relates the results of the study to the description of the business and planned resources, and that also highlights the potential of the business plan.

[(49)]This analysis can also be supported by the use of strategic tools such as SWOT analysis (http://es.wikipedia.org/wiki/An%C3%A1lisis_DAFO) or Porter's Five Forces (http://es.wikipedia.org/wiki/An%C3%A1lisis_Porter_de_las_cinco_fuerzas).

3.2.7. Marketing plan

The purpose of the marketing plan is to define the commercial strategies that will enable us to reach the predicted turnover of the financial analysis, which we will look at in detail in the following section.

Hence, the marketing plan details the actions we need to take in order to apply the business model and opportunity described in the business plan and to exploit their competitive advantages.

Thus, the marketing plan needs to take into account the following:

- **Overall commercial strategy**. The overall strategy needs to define how the sales aspect is integrated into the business project. We need to explain how we will identify clients and how we will contact them, why will clients be interested in or decide on the products or services we offer, and hence, which features of our products or services we will emphasise to generate sales, such as price, quality, guarantee, technical support, etc.
  Free software is a prime example of this, since its main attraction for potential clients is the reduced costs it generates rather than its quality, which is often superior to that of proprietary software. In contrast, clients usually identify the high prices of proprietary software with superior quality, and free software, which is more economical, with inferior quality. Hence, when drawing up a business plan based on a free software business model, it is essential to emphasise the superior quality of[50] free software.

(50)This can be done by emphasising the interoperability and flexibility and the constant revisions and improvements undergone by free software.

- **Sales strategy**: defines the short- and long-term sales aims and the market sectors in which the products or services offered will be introduced initially and in the long run. In all events, decisions must be justified and backed up by the results of the market study.

- **Price strategy**: first of all, we must determine the prices at which the products or services will be sold, comparing them, if possible, with those of the competition, estimating a gross profit margin and evaluating whether this is sufficient to sustain the company's entire business activity[51].
  It is very important to justify the price policy, especially when comparing it to that of the competition. If the price of the products or services offered is higher than that of the competition, this should be explained in terms of innovation and quality, features and enhanced guarantees. If the price is lower, we will need to explain how we will make it profitable (for instance, greater efficiency and lower production costs). Again, it is very important to explain the reasons for the low cost of free software and the benefits that this brings.
  Lastly, the price strategy should be optimal, i.e. it should maximise the profit margin, and hence, profitability. A higher price can sometimes generate greater profits, even though it partially reduces sales.

(51)It is also a good idea to compare your own margins with those of the competition, if you have access to this information.

- **Sales policy**: this determines the composition, form of contract and profile of the sales team, including representatives, at the launch of the company and in its medium- to long-term evolution. It includes the sales margin policy and the promotional measures that will be offered to sales representatives and authorised distributors.
  The sales policy also includes: estimated sales for each sales representative, incentives, the collection periods agreed with clients, and special promotions such as discounts, advances, rebates, etc.

- **Promotion and advertising**: you should describe the measures that will be taken to attract the attention of potential clients to the products or services offered. These measures include mass e-mailing, participation in trade fairs and events, advertising on websites, etc.
  We will eventually need to quantify the cost of the promotion and its return through consultations with clients and the sales we have closed.

- **Aftersales service and guarantees**: you should describe the aftersales service and the guarantees of the products or services, where applicable. In other words, what type of service and guarantee are offered, their duration and price (if optional) and their cost for the company.
  With free software, part of the aftersales service is indirectly provided by the community of developers and users, which constantly improve successful products. The free software company plan should keep this in mind and explain it as an advantage, but never as the only form of additional support. Remember that the vast majority of clients would like the aftersales service to be included and guaranteed in the conditions of sale.
  Lastly, we also need to assess the impact of our aftersales service and guarantees on the client's final decision and compare our service with that of the competition.

- **Distribution policy**: the distribution policy should describe the distribution channels that will be used and the discount, commission and margin policies applied to each of these channels.
  In free software business models, we often come across programmes for partner companies[52] in different forms: system integrators, software vendors, etc., and those who are paid commission and offered dedicated services and assistance and access to promotional channels.
  As we explained above, free software products and services can easily be offered on the global market, so the marketing plan must study this possibility and its potential features, including the effect of international laws on the company's activity, overseas collection management, etc.

(52) One example is the Openbravo free software ERP partner programme, which you can visit at http://www.openbravo.com/partners/join-openbravo/details/.

3.2.8. Financial analysis

The financial analysis or study is also one of the essential parts of any business plan, since its aim is to evaluate the feasibility and financial potential of the business project, detect the investment needs for its launch, identify the resources available initially and describe the various possibilities of financing.

Contrary to what we may think, the financial analysis is one of the most creative parts of drafting a business plan.

The financial statements or main headings that need to be covered by the financial analysis are as follows:

- Cash position over the first year, broken down by months to reflect the effects of seasonality[53].
- Analysis of working capital, which allows us to determine the liquidity of the company.
- Calculation of the balance point and alternatives if target sales are not reached.
- Financing needs and alternatives, selecting those that are most profitable and including elements to explain the decision.
- Annual balance sheets with a five-year view and the first year broken down by months.
- Source and application of funds, allowing us to predict risk situations for the company and evaluate the source and long-term use of funds.

(53) Even highly technological business plans, such as those based on free software, can be affected by economic seasonality (the summer holiday period, for instance).

Working capital

Working capital measures an organisation's balance of assets and liabilities and confirms that there are more liquid assets than short-term debts. For more information, see http://www.innovaceei.com/es/knowledgebase/index.asp?faqsRecid=385&faqRecid=385&show=4460.

Combined analysis of these financial statements is recommended as we can draw conclusions about the business project as a whole: the amount of capital required and when it will be required, and the necessary debt and when it should be paid, among others.

We should also explain the expected return on our investment and indicate when this will be recovered.

As explained above, we need to avoid falling into the trap of presenting an overly optimistic financial analysis to investors in order to win them over, since this will go against the company sooner or later and question marks will be raised about its viability and credibility.

3.2.9. Legal form

If the last aim of the business plan is to create a new company[54], we need to choose the legal form of this new company, its tax system and its founding partners. We will also need to indicate the name of all partners and investors together with their participation in the new society.

(54)If the business plan is for an existing company, this section should describe its legal nature and any modifications that implementation of the business plan could bring about.

It is a good idea to detail the procedures required to set up the new company step by step, together with their costs and the time needed for them. We must also indicate whether we will be using the services of external advisory specialists and the cost of these.

3.2.10. Risk management

All business projects, whether to create a new company or a new line of business, involve numerous risks that are sometimes unavoidable. Hence, the business plan should offer a complete description of the risks and their consequences.

More information

You will find a general introduction to this topic in the "Risk management" section of the first unit.

Risks can be classified as internal (originating in the company) or external and by the functional area that they affect: technical, commercial, etc.

For example, internal risks can include delays in production or a lack of qualified staff, while external risks can be a new market regulation that partially reduces return or the emergence of new technologies that cause the products or services offered to become obsolete.

We need to define a contingency plan for each risk, which includes a series of preventive[55] actions, i.e. measures to try and prevent the risk from occurring, and a series of actions to mitigate or remedy[56] the risk, which should be adopted if it materialises.

(55)For instance, to prevent the appearance of new technologies that could leave the products or services in the business plan obsolete, we should practice active technological surveillance, possibly collaborating with companies or organisations that work in the same area.

Some risks can have negative effects, but they can also be positive. For example, changes in the legal or political framework that can affect the business model but which also provide new business opportunities.

(56)For example, human and material resources for other departments could be used to recover a delay in production.

The correct identification and assessment of risks in a business project and the drafting of suitable contingency plans for them, far from highlighting project weaknesses, actually emphasise the management skills and precaution of the business developers and enhance their credibility.

3.2.11. Summary and evaluation

The last section of the business plan should summarise the strengths and weaknesses of the business project, the advantages and opportunities it offers and its main risks and threats.

The summary is your last chance to persuade a potential investor so you need to be very convincing and seize the opportunity to emphasise the arguments in favour of the business project and those that its developers believe in.

However, after drafting the business plan, the project developers may find that it will not be as profitable as they had hoped or even discover that it is completely unfeasible. This shows how useful the business plan is as a tool to identify the best business opportunities.

3.2.12. Business plans and free software

Drafting a plan for a free software business is not that different to the procedure for drafting business plans in other sectors and we have already seen some of its features in the previous sections.

In general, we need to remember that a business plan may address different types of reader: advisors, investors, technicians, bankers, etc. Therefore, we need to use a language that they can all understand and avoid using highly technical vocabulary. When the use of technical terminology is unavoidable, you should explain each concept clearly in simple terms. Investors never invest in anything that they do not fully understand.

We also need to take the time to explain the features of free software, pointing out the differences between it and proprietary software, and highlighting its main advantages. Do not hesitate to use real-life examples and examples of success to back up the arguments put forward in the business plan.

Although free software is adopting an increasingly relevant role in the media and society thanks to the commitment of the free software community, companies and government bodies, its nature and financial implications are not so well known.

Again, we must take the time to explain the free software business models carefully and be prepared to answer and possibly even anticipate the most common questions, such as, How can you invest in and earn money from something that anybody can copy?

3.3. Production of free software

Many of the business models described in the first section depend to a greater or lesser extent on free software development.

One of the problems with free software projects is that only the successful projects are echoed in the community and only the very successful ones reach the non-specialist media.

However, before we turn to look at free software production, we should remember that the vast majority of free software projects are a failure for specific reasons. It may simply be that the project fails to produce quality and competitive software or that it does not manage to attract the attention of the community of developers and users.

Needless to say, as we saw in the subject materials, a free software project should be dealt with as a software project and only in the last instance, as a simple engineering project. Hence, a free software project will pose the same initial risks and problems as any other project.

However, given the free nature of this type of project, there are other strengths and weaknesses to take into account. Due to the seemingly non-professional nature of many free software development projects, their execution may appear easier than traditional software development projects, but there can be nothing further from the truth.

The aim of this section is to describe the features of free software development projects, contrasting them with proprietary software development and offering a series of good practices to encourage their success. These practices correspond to the key areas and elements required to set up and execute a free software project, namely:

- Creation and presentation of the project
- Necessary infrastructure
- Organisation of the community
- Development
- Releasing and packaging
- Choice of licences

Naturally, not all of these steps are compulsory. As we saw in the business models, a free software company might initiate a project or, as occurs in most cases, it might join an existing project.

This last option is often the most advisable and, given the nature of free software, it does not rule out the possibility that a new project could be created from an existing one under the identity of the company or organisation interested in leading the development.

3.3.1. Creation and presentation of the project

This section deals primarily with the steps required to create a new free software project and present it to the community.

Thus, the first step to take before creating a new project is to find out whether there is a project that already does what we propose, at least in part. If there is a similar free software project that we can contribute to or reuse to launch a new project, we can contact its leaders to explore the possibilities of collaboration and their future plans.

Generic search engines

Generic search engines are the first step to finding existing projects, along with news sites, directories and public forges, such as http://freshmeat.net, http://directory.fsf.org and http://www.sourceforge.net.

If we decide to create a new project, the first thing we should do is choose a name that will identify it in the community. Generally speaking, a good name will give an idea of what the software does or at least its field of application, and it should be easy to remember.

Whether we like it or not, English is the de facto official language of the Internet. So, if we want our project to have a global impact – and this should usually be the case – we should try to come up with a name that will have some meaning in English or that is neutral[57].

(57) In other words, a name common to several languages, such as Apache, or which is not associated with any major language, such as Ubuntu.

We should also pay attention to legal aspects, to ensure that the name does not conflict with brands and that the associated high-level Internet domains[58] are still available.

(58) In other words, .com, .net and .org.

As we saw in the section on the creation of business plans, all projects must have a clear definition of their mission that will attract the attention of users and developers and let them decide whether or not they are interested in the project.

Along with our mission, it is important to state clearly that the project concerns free software, which means making a clear reference to *free software* or open-source software).

Other key elements in the presentation of a free software project are:

- **List of planned functionalities[59] and current requirements**. This should be drawn up in simple terms without the use of technical vocabulary. It is a sort of detailed summary of what the software does and allows users to find out easily whether it has the functionalities that they are looking for.

(59) These can be indicated along with the words "in progress" or "in development", ideally with the date or version when they will be available.

The requirements should also be easy to understand so that users know whether the application can be installed and used on their system.

- **Development status**. In the free software community, users are usually very interested in knowing how the project is coming along, both if it is a new project and if it is an older one. Thus, we should explain the short- and long-term aims of the project, and the functionalities currently being developed and that will be available in future releases, etc.

- **Available downloads**. The source code should always be downloadable in standard formats, using a straightforward method that does not complicate the process for the user[60].
  The installation process should also be simple and, most importantly, comply with the standards from the very start. It is not initially necessary to provide binary packages or executables unless the compilation process is very complicated.

(60) For example, it is preferable to avoid user registration processes for access to the download area.

- **Development repository**. Unlike users, potential developers are more interested in accessing the working repository, where they can follow the day-to-day evolution of the project and participate in it, either by adding new functionalities or fixing bugs. Thus, it is a good idea to allow everybody anonymous read access to the repository.

- **Bug tracking**. As with the repository, the bug tracking[61] database should also be open to everybody. Paradoxically, the more bugs the project database contains the better because this means more users and more participation in the project.
  There won't be many bugs at the start of the project. It is good practice to log any bugs fixed internally by the project team in the database.

(61) We often come across the terms *bug tracker* and *bug database*.

- **Communication channels**. One of the aims of any free software project is to create a community around it and, in order for this community to organise itself, we need to facilitate the right communication channels. This includes mailing lists, IRC channels, forums, etc.
  In the first phase of the project, it is wise not to diversify or specialise the communication channels too much. A single forum or distribution list for users and developers may be sufficient to encourage interaction between them.

- **Documentation for users and developers**. Documentation is essential for any free software project, for both users and developers.
  Good user documentation needs to explain how to install the software and how to use its functions. You can also provide users with a basic tutorial, containing a step-by-step explanation of how to perform the most common tasks. Maintaining a section of frequently asked questions or FAQs is the perfect complement to the documentation.

Developer documentation should include the contact details of the main project developers, instructions for sending error reports and patches, and a presentation of the development organisation and the decision-making process used by the developers.

We will look at all of these elements in detail in the following sections.

To conclude this section, we would like to emphasise the importance of appearance – that is, how the free software community sees the project – for the success or failure of a free software project.

Many developers neglect this communicative and public relations task, but it forms an essential part of virtually any successful free software project.

For this purpose, we will need to clearly define the aims of the new software, which can usually be summarised as:

- **To explain clearly what the software does**: its main functionalities, the current state of development and future plans, and its positioning vis-à-vis existing solutions and projects.

- **To raise the profile of the software**: ensuring that it reaches the community or market of potentially interested users and developers.

- **To promote the use of the software**: ensuring that potential users and developers know how to use the new software and adopt it instead of the alternative solutions.

- **To involve new developers in the project**: allowing them to contribute to the development of the project through the implementation of new functionalities and to state their opinion on the future direction that the project should take.

These last two aims, obtaining lots of users and lots of developers, are often the most important ones. However, we need to implement one strategy for users and another for developers since, while they form part of the same free software community, they represent two very different audiences.

We need to clearly define the message we wish to convey to each and structure it with a gradual complexity to ensure that the level of detail corresponds to the effort required by the reader. For example, there is no sense in saturating the user with software architecture or explaining technical details to the developers without first giving them a reasonable overview of the architecture.

Finally, this message should be easily accessible, reaching its audience through advertisements on forums or related communities, on the project website or even in the documentation, among other options.

3.3.2. Infrastructure

All free software projects need a series of tools to manage the information generated daily by the project, from the developed code to communication among its members.

We introduced some of these tools in the previous section because they are needed to implement the project:

- **Website**
  This provides a centralised source of information about the project and offers access to other specialist management tools.

- **Mailing lists**
  This is one of the most common channels of communication in free software projects. Message exchanges are usually archived and used as reference and to form a knowledge base for the project.

- **Version control system**
  This allows developers to control the creation and management of the code, returning to previous versions and merging different versions. With the version control system, anybody can visualise the current status of the code and its evolution over time.

- **Bug tracking system**
  This allows developers to track the functionalities and bugs they are working on individually and to coordinate themselves and plan new releases. Although bug tracking is its main function, the database can also be used to track any project task, such as new functionalities.
  With the bug tracking system, anybody can find out if a bug has been fixed or if it is being worked on. In conjunction with the version control system, it tells us about the dynamism and logged activity of the project.

- ***Chats***
  These are a communication channel for solving queries and problems quickly. Conversations are not generally archived so it is better for complex discussions to take place on mailing lists.

Each of these tools responds to specific needs, mainly connected with communication and information management. The experience and characteristics of the community of users and developers associated with the

project will dictate the configuration and use of these tools. Nonetheless, it is worthwhile pointing out a few aspects that could be useful in most free software projects.

Mailing lists are an essential part of any free software project so we need to pay special attention to our management and use of them. It is virtually a must to have a management system for distribution lists, whose configuration and maintenance could be complicated in the early stages.

Internet resources

The most popular systems include Mailman (http://www.list.org), Smartlist (http://www.procmail.org), Ecartis (http://www.ecartisorg), Listproc (http://listproc.sourceforge.net) and Ezmlm (http://subversion.tigris.org/hacking.html).

The main options and functionalities of a distribution list management system are as follows:

- Subscription by e-mail or a web interface
- Subscription in *digest* or normal mode[62]
- Moderating
- Administrator interface
- Configuration of message headings
- File management and querying

(62) In *digest* mode, subscribers receive a regular compilation of all messages, usually once a week or once a month, while in normal mode, the messages are received immediately.

Mailing lists can also be integrated into other tools, such as the version control system or the bug tracking system, to inform of aspects such as source code changes or modifications to the error status or tasks in course.

The version control system is also essential for any free software project that hopes to create a developer community. Almost all version control systems operate through the existence of a remote copy, common to all developers, whose versions can all be consulted. Each developer has a local copy of this remote copy that he or she works on. Occasionally, the developer sends his or her modifications to the remote copy to share them with others.

The main functionalities of version control systems are:

- **Committing**: integrating the changes from the local copy to the remote copy, which will then be logged in the version control database.

- **Updating**: integrating the changes of the other developers in the local copy.

- **Checking out**: obtaining a local copy from the remote copy.

Any document or file edited in the project can and should be subject to version control, which should not be limited to source code files. The use of a version control system can be very practical for editing and sharing documentation and technical reports and generally any document created and maintained jointly.

As explained above, the bug tracking system has many other functions besides that suggested by its name. These include the tracking of all types of task, such as the implementation of new functionalities, the preparation of releases and user support.

The life cycle of a bug is usually as follows:

- **The bug is reported**: All bugs include at least a summary and initial description containing, where possible, the elements needed to reproduce it. Most bug tracking systems allow us to set up specific fields. Remember that bugs can come from both the community of users and the community of developers.
  Once archived, the bug status remains open and is not assigned to anybody. During this time, the individuals who access the database can read the bug description and ask the user or developer who reported it for more information.

- **The bug is reproduced**: based on the instructions in the bug description, somebody manages to reproduce the bug, thus validating it. In other words, we can now say that the bug is real.

- **Bug diagnosis**: in the previous phases, a developer takes responsibility for fixing the bug or somebody in an authoritative position in the project assigns it to the most suitable developer.

- **Bug assignation**:
  this must be entered in the database so that we do not have more than one developer working on fixing the same bug without realising it. It is also possible to report the expected fixing date or release in which the bug will have been fixed.

- **Bug fixing**: once the developer has fixed the bug, he or she will mark it as fixed or closed.

Bugs can sometimes be fixed quickly, so some of these phases may not be necessary. Sometimes, the bug is not really a bug and may simply be caused by incorrect use. In all events, no matter how easy the solution is, it is always a good idea to log the bug and report it correctly to users.

Another common situation is when several users report the same bug, referred to as duplicate bugs. In this case, it is best to group all of the reports into one so that we can concentrate our efforts and put all of the information in the same place.

Lastly, a bug may be reported as fixed when it has not actually been resolved, generally because the steps followed to reproduce it do not match those indicated by the user who reported the bug. In this case, the user can reopen the bug, adding all of the necessary information. There are numerous public forges offering these and other tools, ready for use in free software projects. These platforms come with a series of advantages and disadvantages.

Internet resources

The most popular public forges include SourceForge (http://www.sourceforge.net), Savannah (http://savannag.gnu.org and BerliOS.de (http://www.berlios.de). Some organisations also offer hosting for projects in their area of interest, such as Apache (http://www.apache.org) and Tigris (http://www.tigris.org).

Their advantages include their capacity and available bandwidth: the success of the project is irrelevant because the servers will always be in operation. Keeping a high-availability server running requires a lot of extra work. Moreover, the tools available on these forges have already been configured and are usually very easy to use. Obviously, the main disadvantage is the limited flexibility and configuration possibilities of the tools.

So, before starting our project, we may want to host it on a public forge but be open to the possibility of our own hosting in the future, starting by registering the name of the domain associated with the project. For example, while not the perfect solution, having a website informing about the project that redirects to a public forge for aspects of code development can be a good compromise.

3.3.3. Organisation of the community

One of the biggest differences between free software projects and proprietary software projects is the way in which the developer community is organised.

In proprietary software projects, the structure is normally that of a hierarchical organisation of the team or department in charge of development in the company. Although hierarchies can sometimes be seen in free software projects, partly due to the merits of the individual developers, the organisation of the developer community is more flexible and also stronger.

Paradoxically, one of the reasons why the developer community works as one and remains close-knit is the possibility of creating a new independent[63] project from the original project. The possibility of a free software project

(63) This is known as forkability, i.e. the possibility of *forking*.

forking is usually negative for both developers and users. It is precisely this threat that makes the community organise itself and strive to ensure that decisions are taken as a group.

In other words, the possibility of forking makes the community strive to achieve a more or less democratic consensus in major project decisions.

There are generally two forms of organisation for free software communities, although most projects end up adopting a position midway between the two. They are:

- **Organisation based on a "benevolent dictator".**
  A benevolent dictator is a figure of authority who makes final decisions with repercussions for the development of the project. Nonetheless, benevolent dictators often do not make decisions directly but act as moderators in discussions, attempting to conciliate the viewpoints of the developers and identify the most valuable contributions. Another of the actions of the benevolent dictator is to delegate experts to deal with the decisions or discussions underway. Benevolent dictators are usually developers with sufficient experience in the project and related technologies. However, they do not need to be the most expert developers; they must simply be capable of understanding the project as a whole and recognising the best contributions.

- **Organisation based on consensus.**
  The term consensus is used to refer to agreements accepted more or less tacitly by the entire community, i.e. where nobody opposes the decisions or the direction taken by the project. As a result, the process of consensus is not usually formal by any means. However, if a consensus cannot be reached on a given issue, a vote can be taken.
  Most discussions that take place during the development of a project are usually technical, so consensus is achieved when everybody agrees on an issue, such as the design or implementation of a functionality or the way to fix a bug. In these cases, a member also usually summarises the discussion at the end.

Generally all communities, particularly those based on consensus, have excellent support in the version control system, which means that they can go back and undo any decision that turns out to be incorrect.

Projects usually begin with an organisation based on a benevolent dictator and evolve towards an organisation based more on consensus as the community expands. This usually occurs at certain times in the development of the project, such as when a benevolent dictator gives up his or her position and the authority is distributed across the community, particularly among its most highly regarded members.

After a time, the conventions and agreements adopted through consensus by a community can become increasingly large, so the main points should be set down in a document that can be used as a guide and for future reference. This may include both the form of governance of the community and the conventions and recommendations for developers.

Internet resources

Take a look at the guides for the Subversion project (http://svn.collab.net/repos/svn/trunk/HACKING) or the Apache foundation (http://www.apache.org/foundation/how-it-works.html and http://www.apache.org/foundation/voting.html).

Lastly, we need to ask ourselves what role can companies play in free software communities.

On the one hand, we could have a company that wishes to start up a free software project and create a community of users and developers. And on the other, we could have a company that joins a free software project already underway. In both cases, the company must clearly define its aims with this free software project and know what its participation in the community will be.

There are many possibilities. For example, the company may seek a leadership position in the community and lead the project, or it may simply take part in discussions, participate actively in the implementation of new functionalities or have just a selection of its developers deal with the problems of its clients.

Bearing in mind the difficulties of constructing a successful free software project, it is clear that, at least in the case of projects begun by companies, the community already exists: it is formed by the developers in the company and its clients.

In these situations, the benevolent dictator model will probably be the most appropriate form of organisation, to start with at least, but we will still need to define the rules for participation in the community. The challenge lies in converting these clients into active users who will help to improve the project and in getting other developers involved.

The solution, albeit a difficult one, is to offer benefits or some form of added value to the users and developers who participate in the community.

It is good practice for the company's development team to be fully integrated in the community and to follow the development methodology for the free software project. This means that the developers must participate in the project over a long period of time in order to become familiar with the operation of the community and gain credibility in it.

3.3.4. Development

This section describes the development process for free software projects, not from a technical perspective, as this will depend on the nature of the individual projects, but from the point of view of project management and developer coordination.

With development, we need to remember that one of the differences between free software projects and proprietary software projects is the absence of a centralised organisation. For example, when the date of a new release draws near, a company can assign a certain number of resources to prepare for it. The voluntary developers forming the community, on the other hand, are not so easy to direct. Their individual reasons are different and while some may be interested in publishing a new release on time, others may only be interested in a specific functionality.

Thus, the distribution of tasks in a free software project is based mainly on the independence between them, with the general rule being that each developer works on what he or she wants when he or she wants.

Yet this approach is ideal in part, and a person or team is needed in most free software projects to coordinate all the voluntary developers. This team can be formed explicitly by the initiators of the project or the benevolent dictator or implicitly by members with more experience and influence over the community.

Some of the basic tasks of coordination, required to carry out the project successfully, include:

- **Delegating**: one of the main tasks of the project coordinators is to delegate tasks to other developers. When somebody delegates a task to another person and the latter accepts it, the benefits are two-fold: the coordinator finds somebody to do the work for him or her and this person's efforts are acknowledged in the sense that they have been entrusted with a task. Hence, the best way to delegate a task is through a channel of communication that can be seen by the entire community, always giving the option of turning down the offer.
  In this case, the coordinator must be aware of the skills and interests of the community members and direct the offers on this basis. For example, there is no sense in asking somebody to do something if they lack the necessary skills or if they already have several ongoing tasks.

- **Criticising and praising**: the correct evaluation of contributions from each project developer is very important for the creation of

a friendly atmosphere within the community. Moreover, evaluations from coordinators or higher members have greater repercussions on the community.

Hence, both criticism and praise should be used wisely. Continuous or unfounded criticisms will no doubt provoke negative reactions, as will the same sort of praise. However, in a technical discussion, detailed criticism can be constructive because it means that the person who is making it has taken the time to analyse the design or implementation being criticised.

- **Avoiding territoriality**: one situation to avoid is where some members of a community attempt to appropriate part of the project ("their part") and refuse to accept criticisms or contributions from other members. Although this attitude may appear positive at first, since these members are usually experts and spend a great deal of time on their part of the project, the long-term result is that no other developer revises the code, which can mean a loss of quality and lead to fragmentation of the community.

- **Automating tasks**: most developers generally work on one part of the code and do not know what the others are doing. Therefore, the coordinators must make it their task to obtain an outline of the project and be aware of what each member is doing. It is easy to identify a series of tasks inherent to code development that all developers must carry out and which it is often useful to automate and centralise.

  The clearest example of this is the automation of tests[64], which allows developers to make changes and experiment with parts of the code that they are unfamiliar with.

- **Treating users properly**: the existence of an active user community contributing valuable information to developers is key to the success of any free software project. However, developers and users often speak different languages, to put it one way. Many users are not familiar with software development or how free software communities operate. Developers need to be able to put themselves in the place of users and try to explain themselves as clearly as they can.

  Just remember that any user could be a contributor to the community and because the vast majority of users never address the developer community, we need to reserve special treatment for those who do. For example, when a user indicates that the documentation is incomplete, we can suggest that they complete it themselves, or when they report a bug, we can ask them to try and fix it. And of course, always thank them for their contribution, whatever it is.

- **Sharing management tasks**: besides code development, every project has a series of management tasks that become increasingly complex as the project expands. The coordinators or team that initiated the project usually take responsibility for them, but it is good to share them with other

(64) Specifically, we can create a test package, a programme that runs the project software to reproduce all previously known and fixed bugs. This allows developers to make sure that they do not reintroduce old bugs that have now been fixed.

members of the project, as we saw in the section on delegation. These tasks include:

– **Patch management**. Controlling which patches have been received and analysing them for acceptance or, as is usually the case, to detect their problems and report them to the author of the patch.

– **Translation management**. Coordinating the translation of the documentation and software.

– **Documentation management**. Keeping the documentation and the frequently asked questions or FAQs section up to date, introducing changes as they appear.

– **Bug management**. Managing the bug database, which includes ensuring its integrity and checking for duplicate bugs, among other functions.

- **Permissions management**. One of the more important management tasks is the management of permissions, i.e. deciding who has permission to commit and hence, integrate their code into the remote copy in the repository. Besides granting permissions, there is also the possibility that we will have to revoke them.
  Developers who do not have permission to commit can of course still contribute to the development of the project by producing patches to fix bugs or add new functionalities that will be analysed by the project developers and eventually incorporated. In fact, the most common mechanism for obtaining permission to commit is for a developer to contribute patches to the project until the team of developers considers that his or her contributions and knowledge of the project are valuable enough.
  To encourage the participation of new developers, it is useful to make the procedure for obtaining permission to commit and the procedure for withdrawal of this permission public and as transparent as possible.

3.3.5. Releasing and packaging

Preparing releases and packaging is, besides code development, one of the most important tasks of the entire free software project.

A new release involves changes, particularly for users. Firstly, all known bugs from the previous release will have been solved and it is highly likely that there will be new ones. There may also be new functionalities and configuration options. There may even be incompatibilities between the new version of the software and the previous ones, in the format of the data, for instance.

Since the jump from one release to the next can have important consequences – and not all of them good – one of the first aspects we need to decide on is how to identify each of the releases.

There are several conventions for this, some more creative than others, but the most common method is to number them with a series of digits separated by decimal points. For example:

- Release 3.4.1
- Release 3.4.2
- Release 3.5
- Release 4.0

The meaning of these digits can vary. Changes to the third digit usually indicate fixed bugs or minor improvements to some functionalities. Changes in the second digit usually indicate the introduction of new functionalities. And lastly, changes to the first digit indicate new groups of functionalities and no doubt important changes in version compatibility.

It is a good idea to indicate the meaning of the numbering of the releases on the project website.

Website

See the versioning of the APR project (http://apr.apache.org/versioning.html).

In addition, some releases are usually identified with the words *alpha* or *beta*, indicating their development status. For example:

- Release 3.4.1 (alpha 1)
- Release 3.4.1 (alpha 2)
- Release 3.4.1 (beta)

Generally, the word *alpha* is used to designate the first release, which allows users to access the software and all of its functionalities but for which a considerable number of bugs is expected. Users who install and run an *alpha* version usually do so to evaluate the software and report bugs to the team of developers. A *beta* version, on the other hand, has undergone far more debugging and, if it contains almost no bugs, will become the official version: this is what is known as a *candidate version*.

For developers, a free software project is in a constant releaseprocess and they always use the latest version available in the repository for development, so it can be difficult to specify the exact moment of the release.

The best practice in this case is to keep a branch in the repository containing the code that will be introduced in the next release, regardless of the trunk. This also ensures that the developers not involved in preparing the release can continue to work on the project.

Thus, one of the most important parts of the process for preparing a release is its stabilisation, i.e. deciding which changes and functionalities will be integrated into the branch of the next release. Here, the decision-making mechanism of the free software community should come into play, giving us two basic alternatives:

- Designating *a* release owner that will decide which changes to introduce in the future release.

- Voting for the changes to be introduced in the future release, for which we will need to define the voting rules. An intermediate solution is to establish the minimum number of developers needed to vote for a certain change in order for it to be included.

In addition, one or two release managers can be appointed to integrate and validate the changes to the branch of the release.

Free software is usually distributed as source code, adequately packaged and compressed in a standard format. The name of the package is usually formed by the name of the package, the version number and the appropriate suffix for the format. For example:

- myproject-3.4.1.tar.gz
- myproject-3.4.2.zip

The information that should accompany any new release includes the licence under which it is distributed, the instructions for installation and set up, and the changes and features added since the last release. This information is included in a series of files with more or less standard names: *LICENCE* or *COPYNG*, *README* or *INSTALL*, and *CHANGES*.

And finally, users need to compile the source code and install it on their system, which should always follow a standard procedure in order to reach as many users as possible. Another possibility, used especially with mature software, is the distribution of binary packages, either as executables or installables, which does away with the need for users to carry out the compilation process manually[65].

(65) For example, the RPM or DEB system with GNU/Linux systems and MSI or self-installing executables in Windows.

From the perspective of free software companies, the releases policy is one of the key tools for reaching potential users of the software. Correct planning of releases should meet the user needs of the moment, whether by adding new functionalities or by fixing bugs, so we must determine a suitable rate of publication for new releases.

For instance, publishing new releases too often can saturate users and they will probably not install all of the releases. In contrast, leaving too big a gap between releases could encourage users to look elsewhere for solutions. It is also a good idea to guarantee the quality of the new releasesby trying to fix as many bugs as possible before their publication. The effect of a release plagued by bugs gives a very poor image of the project and the company behind it, which can be difficult to remedy afterwards. It is therefore very important to focus on the open and cooperative nature of free software in order to improve the quality.

3.3.6. Choice of licences

The differences and advantages and disadvantages associated with the various free software licences make this one of the most hotly debated topics. One thing is certain though and that is the choice of a particular licence plays a minor role in the adoption and success of the project as long as it is a free software licence. The vast majority of users choose their solution based on the functionality and quality it offers, rather than its licence.

The most important thing is to be clear on the project aims and the free software company's aims for the project. On the basis of these, we should choose the most suitable licence or create a new licence based on existing ones[66].

(66) Appendix I includes a short list of the main licences used in the production of free software.

Many free software projects have their own licences, adapted to their needs and aims[67].

(67) Examples include the OpenBravo licence (http://www.openbravo.com/product/legal/license/) or the dual MySQL licence (http://www.mysql.com/about/legal/licensing/).

Free software licences and the relationships and potential incompatibilities between them can be very complex and we sometimes have to call on lawyers or specialist legal experts for help.

See also

The *Legal aspects and the features of exploitation of free software* subject of the official Master's Degree in Free Software looks in detail at these issues.

One of the main sources of incompatibility is the reuse of free components under restrictive licences. A typical example is the GPL licence, which requires any software that uses GPL components to be distributed under a GPL licence.

It is good practice to keep an inventory or chart of the external software and licences used in the project, describing the parts of the code that use them.

Summary

Although free software technology is tested, produced and run in a variety of scenarios and despite the fact that we can easily find news on products, events or figures related to free software in the general media, there are still many clichés about its real and effective implementation.

These clichés and misconceptions often have a negative impact on the implementation of free software systems, both in domestic situations and in organisations. Users often write off free software saying that it is for computer experts or *hackers* or that free software applications are unstable, unfinished or lack the necessary support. Business, on the other hand, takes the view that free software does not protect intellectual property sufficiently, that it represents a loss of competitiveness, or that, with certain exceptions, there are no feasible business models for free software.

We can summarise most of these ideas as being concerned with a lack of quality in free software processes and products, particularly in the quality perceived by users. To a certain extent, however, it is fair to say that the history, culture and nature of free software and the community of users and developers have encouraged these ideas.

The growing commitment of governments and major organisations to free software should encourage people to question these misconceptions. Moreover, many experts and analysts have pointed out the potential of free software in bolstering the development of local, European and world economies. For example, Gartner has said that "OSS is a catalyst that will restructure the industry, producing higher quality software at a lower cost".

The materials for this subject attempt to provide a response to some of these clichés, training professionals to carry out projects implementing free software systems through the detailed study – from a conceptual, methodological and practical point of view – of this type of project in a variety of scenarios and situations.

Generally, any free software project must be treated first and foremost as a software project and secondly, as an engineering project, so these materials cannot and should not be a substitute for the necessary knowledge of these topics.

Most of the materials cover the concepts, methodologies and tools required to carry out free software development and implementation projects. This is the reason for the general tone of some of the contents, which gradually introduce the reader into the methodological features of free software projects,

attempting to organise and structure – conceptually and functionally – the main stages and milestones of implementation. We have also explained how business activity linked to free software, whether software development, consulting, integration, implementation or support, can be the object of a profitable, valid and viable business model. Similarly, and although this was not the primary aim of the subject, we have introduced the basic aspects that a business plan should cover, specifically those dealing with free software.

The constant evolution of free software technologies and projects will no doubt make some of these materials obsolete, particularly those on specific projects and solutions. However, both the methodology and the bibliographical references should help us to find the right solution for each project or implementation scenario.

Moreover, it is hoped that the formalisation and structuring of the methodologies and procedures included here, together with the collection of good practices in free software projects will make a qualitative contribution to the community and to the development and expansion of free software in general.

Finally, we could not end without thanking the Fundació de la Universitat Oberta de Catalunya (http://www.uoc.edu) for its support in producing these educational materials. We also encourage all readers who wish to send in their comments or suggestions to contact the authors so that we can improve future editions of this material and the everyday practice of implementing free software systems.

Glossary

business model Business strategy that defines, plans, produces and markets one or more products or services aimed at generating profits for its producers.

direct implementation Process by which the system to be introduced does not require the components involved to undergo complex adaptations or configurations.

free software Series of computer programs and applications whose conditions of use are subject to a free licence.

free software community Group of free software users and developers.

free software licence Licence guaranteeing the four basic freedoms: running the software for any purpose, studying and adapting the source code, redistributing copies of the software, improving the source code and publishing improvements.

infrastructure Series of basic elements or components that are correctly structured, organised and coordinated to facilitate the operation of a system.

insourcing Strategic model consisting of the delegation or production of operations or jobs in an internal department of the organisation, usually specialised, instead of outsourcing them to a third party external to the organisation.

licence Contractual model by which the author of the product (or the copyright owner) sets down the rights and duties of the users of the product and the scenario in which they can use the product.

life cycle of a project Process that covers, structures, organises and coordinates all of the stages and phases guiding the execution of a project and allows us to deal with the complexity of the aims reducing the risk of failure.

market study Analysis conducted as part of a project for a business initiative with the aim of obtaining an idea of the commercial viability of a business activity, based on the general context, competition and consumers.

methodology Systematic analysis or study of the methods and procedures that are, have been or can be applied to a given discipline.

migration Process to substitute infrastructures based on proprietary software for others with equivalent functions based on free software.

open standard Format or protocol subject to public use and evaluation that does not depend on closed formats or protocols. It does not contain clauses limiting its use, is managed independently of specific interests and is available in various implementations (or in an implementation with fair access).

outsourcing Strategic model consisting of the delegation or production of operations or jobs in an organisation external to the organisation, usually specialised, instead of delegating them to an internal department of the organisation.

packaging Process to automate the installation, configuration and uninstalling of software packages on a computer. GNU/Linux systems in particular usually use thousands of different packages.

project Organised, structured and planned process of managing resources to achieve a specific aim, usually strategic.

project management Discipline that studies the best way to organise and administer the available resources to ensure that all tasks required by the project will be completed by the set deadline with the time and costs defined previously.

release or distribution Process to make available an initial version or update a software product for its potential users.

repository A repository, pool or archive is a centralised location for the storage and maintenance of digital information, usually databases or computer files. In free software development, repositories incorporate a version control system that maintains a log of all work and changes made to the archives (mainly source code) constituting a project and allows different developers (sometimes long distances apart) to collaborate with each other.

risk Likely event that could affect the progress of the project and possibly prevent the aims from being reached on time.

strategic plan of the organisation Series of proposals defining the future aims or directions of the organisation. This is normally developed afterwards in the different areas and functional departments of the organisation.

SWOT Acronym of strengths, weaknesses, opportunities and threats. A SWOT analysis is a strategic planning tool used to evaluate the strengths, weaknesses, opportunities and threats affecting a project.

system Series of independent physical or logical elements or components that interrelate to act as an integrated, functional unit.

system services Series of functions that can be run, with or without user intervention, and which are considered essential for the operation of the organisation.

systems implementation Process through which one or more technological innovations are introduced to an organisation as the result of an action deriving from its strategic plan.

user support Series of services that either integrally or through diverse means of communication offers the possibility of managing and solving all possible issues taking place during the use of a program or application.

version Number indicating the development level of a program or application.

Bibliography

Abella, A.; Sánchez, J.; Santos, R., et al. (2003). *Libro Blanco del software libre en España*. [Consulted on 1 March 2008]. http://www.libroblanco.com/document/1000-2003.pdf

Abella, A.; Segovia, M. A. (2005). *Libro Blanco del software libre en España (II)*. [Consulted on 1 March 2008].
http://www.libroblanco.com/document/II_libroblanco_del_software_libre.pdf

Abella, A.; Segovia, M. A. (2007). *Libro Blanco del software libre en España (III)*. [Consulted on 1 March 2008].
http://libroblanco.com/document/III_libro_blanco_del_software_libre.pdf

Clearly, D. W.; Fenn, J.; Plumer, D. C. (2005). "Gartner's Positions on the Five Hottest IT Topics and Trends in 2005". [Consulted on 20 May 2008]. http://www.gartner.com/DisplayDocument?doc_cd=125868

Díaz, R. M. (2007). *El arte de dirigir proyectos* (2nd ed.). Madrid: Ra-ma, 1995.

Guitérrez, J. D. (2007). *"Metodología para el análisis decisorio de la implantación de software libre"*. [Consulted on 1 March 2008].
http://www.informaticahabana.com/evento_virtual/files/SWL14.pdf

Hecker, F. (1998). "Setting Up Shop: The Business of Open-Source Software". [Consulted on 1 March 2008]. http://www.hecker.org/writings/setting-up-shop.html

Kerchmer, K. (2005). "The Meaning of Open Standards". Proceedings of 38th Annual Hawaii International Conference on System Sciences 2005. [Consulted on: 1 March 2008]. http://www.csrstds.com/openstds.pdf

Open Formats. [Consulted on 1 March 2008]. http://www.openformats.org/

Open Source Initiative. [Consulted on 1 March 2008]. http://www.opensource.org/

Open Standards. URL: http://www.openstandards.net/ [Consulted on 1 March 2008].

Perens, B. "Open Standards: Principles and Practice". [Consulted on 1 March 2008]. http://perens.com/OpenStandards/Definition.html

Qualipso Project. URL: http://www.qualipso.org [Consulted on 20 May 2008]

Sáez, D.; Peris, M.; Roca, R.; Anes, D. (2007). *Migración al software libre. Guía de buenas prácticas*. Instituto Tecnológico de Informática.

SELF Project. URL: http://selfproject.eu [Consulted on 1 March 2008]

Free Software Foundation. URL: http://www.fsf.org/ [Consulted on 1 March 2008]

Vega García Pastor, I. de la (2004). *El plan de negocio: una herramienta indispensable*. Instituto de Empresa.

Various authors (2005). *Migration guide. A guide to migrating the basic software components on server and workstation computers* (2005). Berlin: Bundesministerium des Innern. [Consulted on 20 May 2008].
http://www.kbst.bund.de/cln_012/nn_836802/SharedDocs/Anlagen-kbst/Migrationsleitfaden/migration-guide-2nd-edition__pdf,templateId=raw,

Appendix

Appendix I: Free software licences

There now follows a short list of the main licences used for the production of free software. Some of these are discussed in detail in the materials of the Introduction to free software and Legal aspects and the features of exploitation of free software (Part II) subjects.

GNU/GPL v3

GNU/GPL is an abbreviation that stands for the *General Public License* of the GNU project.

Internet resource

For more information about the GNU/GPL licence, see http://www.gnu.org/licenses/gpl.html.

It has a robust redistribution policy called copyleft, by which all derived works inherit the original licence, even if they have been combined with others. Linking from modules with different licences is not permitted. The protection policy for original copyright, among other rights, means that the GNU/GPL licence is incompatible with other licences, including the original BSD licence and proprietary licences.

Internet resource

You will find a full list of compatibilities at http://www.gnu.org/licenses/license-list.html.

Version 3 of GNU/GPL is not directly compatible with version 2. However, many programs licensed under the second version allow the use of subsequent versions of the licence under the same terms.

GNU/LGPL v3

The abbreviation GNU/LGPL stands for the *Lesser General Public License* of the GNU project, which is a licence derived from GNU/GPL.

Internet resource

For more information about the GNU/LGPL licence, see http://www.gnu.org/licenses/lgpl.html.

This licence was originally created to allow the use, linking and integration of free software libraries with other types of licence, sometimes proprietary, to work around the restrictions of GNU/GPL licences. The practice has led to the licensing of numerous programs, some of which are now widespread. The programs licensed under GNU/LGPL include the OpenOffice.org office automation package.

Version 3 of GNU/LGPL is not directly compatible with version 2. However, many programs licensed under the second version allow the use of subsequent versions of the licence under the same terms.

BSD licence

BSD is the abbreviation of the Berkeley Software Distribution licence of the University of Berkeley.

Internet resources

For more information about the BSD licence, see http://www.debian.org/misc/bsd.license. You can also see an example of a licence derived from the original BSD at http://www.freebsd.org/copyright/freebsd-license.html, called a *2-clause BSD licence* due to the elimination of two clauses of the original licence.

It forms part of a group of licences (*BSD-style or BSD-like licences*, including the *FreeBSD licence*) called permissive licences because their user rights policy is rather unrestrictive. This policy, called copycenter to distinguish it from the copyleft of GNU licences, allows the commercial use of the product, its conversion to proprietary code and linking from modules with a different licence, among other possibilities.

The original BSD licence incorporates an advertising clause making it incompatible with GNU/GPL. The clause was removed in later versions, producing what is known as the *3-clause BSD licence*, compatible with GNU/GPL.

Website

For more information about the MPL licence, see http://www.mozilla.org/MPL/MPL-1.1.html.

MPL 1.1

The abbreviation MPL stands for the *Mozilla Public License* of the *Mozilla Foundation*).

It is a private initiative that is a hybrid of the BSD and GNU/GPL licences. It is considered a permissive semi-copyleft licence because it offers the possibility of establishing proprietary licences from derived works. Linking from modules with different licences is permitted. Article 13 allows the licensing of one or more parts of the code with a different licence, called an alternative licence. Only when the alternative licence is GNU/GPL – or any other compatible licence – will the part be compatible with GPL and be able to be linked with others if they are too.

Apache 2.0 licence

The Apache licence is a licence of the *Apache Software Foundation*).

Website

For more information about the Apache licence, see http://www.apache.org/licenses/LICENSE-2.0.

It is quite similar to the BSD licence and is considered permissive because it allows for the possibility of establishing proprietary licences from derived works and linking from modules with different licences. The use of patents of this licence and the provisions for compensation make it compatible only with version 3 of GNU/GPL, maintaining the incompatibility with the previous two versions.

X11 licence

The X11 licence, incorrectly referred to as the *MIT licence*, is a licence of the *Massachussetts Institute of Technology, MIT*).

Website

For more information about the X11 licence, see http://www.opensource.org/licenses/mit-license.php.

It is quite similar to the 3-clause BSD licence and is considered permissive because it allows derived works to be licensed as proprietary software and can be linked to modules with different licences. It is compatible with GNU/GPL and related to the X.Org project. Hence, some older versions of XFree86 continue to use it while the more modern versions use the XFree86 1.1 licence, which is incompatible with GNU/GPL because of its requirements applying to all documentation containing acknowledgements.

Website

For more information about the X.Org project, see http://www.x.org/.

CDDL 1.0

The abbreviation CDDL stands for the *Common Development and Distribution License* of SUN Microsystems.

Website

For more information about the CDDL licence, see http://www.sun.com/cddl.

It is based on version 1.1 of the MPL licence. The main differences revolve around two aspects:

- The author (or copyright owner) can restrict the legal jurisdiction of the rights and duties of the software users.
- The licence establishes the requirement of identifying all authors who contribute to the modifications made to derived works.

It permits linking from modules with other licences and the licensing of derivatives with a different licence, which can be proprietary. Its intellectual property features make it incompatible with GNU/GPL.

CPL 1.0

The CPL abbreviation stands for the *Common Public License* of IBM.

Website

For more information about the CPL licence, see http://www-128.ibm.com/developerworks/library/os-cpl.html.

Its aim is to promote the development of open source, maintaining the possibility of combining the code with other licences, including proprietary licences, though it does not permit the licensing of derivatives with another type of licence. It also prohibits the derived code from infringing the patents of the original, requiring the payment of any royalties. It is incompatible with GNU/GPL due to its clauses on the legality of derived works.

EPL 1.0

The EPL abbreviation stands for the Eclipse Public License of the *Eclipse Foundation*).

Website

For more information about the EPL licence, see http://www.eclipse.org/org/documents/epl-v10.php.

This is based on the CPL licence and has a permissive policy with a business focus. The main difference between it and CPL lies in the treatment of patent infringement by software contributors. All code licensed under EPL maintains the licence in its derived works. However, it permits the separate licensing of addenda under other types of licence, including proprietary. Linking from modules with different licences is permitted. Its characteristics regarding the permissiveness of derived works and its treatment of copyright make it incompatible with GNU/GPL.

Appendix II: Open standards

Definition

The SELF project defines an open standard as a format or protocol that is:

- subject to full public assessment and use without constraints in a manner equally available to all parties;
- without any components or extensions that have dependencies on formats or protocols that do not meet the definition of an Open Standard themselves;
- free from legal or technical clauses that limit its utilisation by any party or in any business model;
- managed and further developed independently of any single vendor in a process open to the equal participation of competitors and third parties; and
- available in multiple complete implementations by competing vendors or as a complete implementation equally available to all parties.

Website

For more information about the open standards of the SELF project, see http://selfproject.eu/OSD.

However, there is no one definition of an open standard because every organisation establishes a series of characteristics or practices suited to its particular aims. These organisations can be organisations that develop standards, supra-national councils or state governments. Some definitions require publication under *reasonable and non-discriminatory* (RAND) conditions, i.e. not totally exempt from royalties.

Example

Examples of this are the definition of the European Union or the ITU-T (http://www.itu.int/ITU-T/othergroups/ipr-adhoc/openstandards.html).

Other definitions focus more on the characteristics of the process that a standard should follow in order to be considered open, such as the recommendations of the World Wide Web Consortium (W3C), Bruce Perens or Ken Krechmer.

Internet resources

For more information about these processes, see:
http://www.w3.org/Consortium/Process.
http://perens.com/OpenStandards/Definition.html.
http://www.csrstds.com/openstds.pdf.

Organisations

The leading organisations, associations, institutes and consortiums for information technology standards are:

ANSI: American National Standards Institute.

Website

For more information about ANSI, see:
http://www.ansi.org/.

ETSI: European Telecommunications Standards Institute
(http://www.etsi.org/).

FreeStandards (*The Free Standards Group*): independent organisation that promotes the use and acceptance of free technologies through standards.
(http://www.freestandards.org/.)

ICANN: Internet Corporation for Assigned Names and Numbers
(http://www.icann.org/).

IEC: International Electrotechnical Commission
(http://www.iec.ch/).

IEEE: Institute of Electrical and Electronics Engineers, Inc.
(http://www.ieee.org/).

IETF: Internet Engineering Task Force
(http://www.ietf.org/).

ISO: International Organization for Standardization
(http://www.iso.ch/).

ITU: *International Telecommunications Union*, which groups organisations from the private and public sectors to coordinate telecommunications and global services
(http://www.itu.int/).

JXTA (*JXTA Project*): combination of open *peer-to-peer* or P2P standards with open-source Java implementations.
(http://www.jxta.org/)

OASIS (*Organization for the Advancement of Structured Information Standards*): international non-profit consortium that guides the development, convergence and adoption of standards for *e-business*

(http://www.oasis-open.org/).

OpenGroup (*The Open Group*): international consortium of vendors for the neutral advance of technology
(http://www.opengroup.org/).

RossettaNet (*Open e-business process standards*): association to promote open standards in *e-business*
(http://www.rosettanet.org/).

VoiceXML (*Voice XML Forum*): organisation of industries to create and promote *Voice Extensible Markup Language* (VoiceXML)
(http://www.voicexml.org/).

W3C (*World Wide Web Consortium*): global consortium for the promotion of Internet standards
(http://www.voicexml.org/).

WS-I (*Web Services Interoperability Organization*):
(http://www.ws-i.org/).

Open standards

We will now list the main open standards identified in the SELF project (http://selfproject.eu/en/system/files/D1_WP2.pdf).

- **Unformatted text**
  ASCII, ISO8859 (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=28245) and UNICODE (http://www.unicode.org/).

- **Formatted text**
  ODT (*Open Document Text*) and DocBook (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office).

- **Scientific text**
  ODF (*Open Document Formulae*), MathML (*Mathematical Markup LanguAge*) (http://www.w3.org/Math/) and TeX/LaTeX (http://www.tug.org/) and (http://www.latex-project.org/).

- **Images (frames)**
  JPEG (*Joint Photographic Expert Group*) (http://www.jpeg.org/) and (http://www.w3.org/Graphics/JPEG/), PNG (*Portable Network Graphics*) (http://www.libpng.org/pub/png/) and (http://www.w3.org/Graphics/PNG/), PNM (*Portable Any Map*) (http://netpbm.sourceforge.net/doc/pnm.html), GIF (*Graphics Interchange*

Format) (http://www.w3.org/Graphics/GIF/spec-gif89a.txt), BMP (*Bitmap*) (http://atlc.sourceforge.net/bmp.html).

- **Images (vectors)**
  SVG (*Scalable Vector Graphics*) (http://svg.org/) and (http://www.w3.org/Graphics/SVG/), ODG (*Open Document Graphics*).

- **Video**
  OpenEXR (http://www.openexr.com/), Theora (http://theora.org/), RIFF (*Resource Interchange File Format*) (http://msdn2.microsoft.com/en-us/library/ms713231.aspx), AVI (*Audio Video Interleave*) (http://msdn2.microsoft.com/en-us/library/ms779636.aspx).

- **Printing**
  PDF (*Portable Document Format*) (http://www.adobe.com/devnet/pdf/), PS (*PostScript*) (http://partners.adobe.com/public/developer/ps/index_specs.html).

- **Hypertext**
  HTML (*Hyper Text Markup Language*), XHTML (*Extended Hyper Text Markup Language*) (http://www.w3.org/MarkUp/).

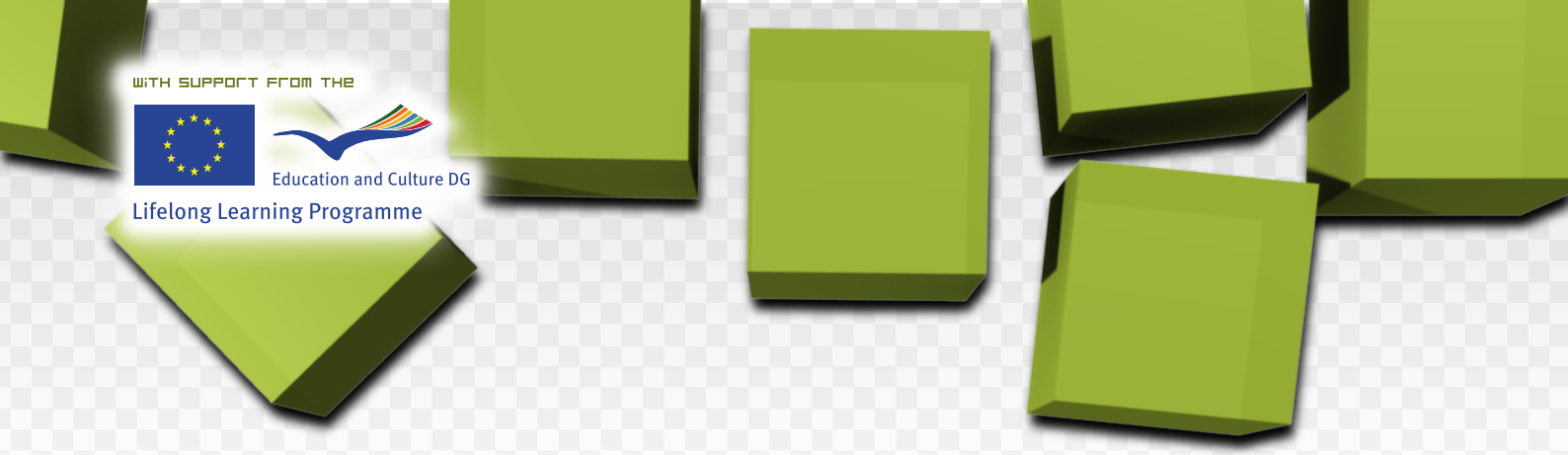
- **Presentations**
  ODP (*Open Document Presentation*).

- **Audio**
  Vorbis (*OGG Vorbis*) (http://www.vorbis.com/) and (http://xiph.org/), FLAC (*Free Lossless Audio Codec*) (http://flac.sourceforge.net/)) and (http://xiph.org/), RIFF, WAV (Wave) (http://www.borg.com/~jglatt/tech/wave.htm).

- **Education and learning**
  LOM (*Learning Object Metadata*) (http://zope.cetis.ac.uk/profiles/uklomcore), SCORM (*Sharable Content Object Reference Model*) (http://www.conform2scorm.com/), IMS (http://www.imsglobal.org/commoncartridge.html), LD (*Learning Design*) (http://www.imsglobal.org/learningdesign/index.html).

- **Business**
  XBRL (*Extensible Business Reporting Language*) (http://www.xbrl.org/).

WITH SUPPORT FROM THE
Education and Culture DG
Lifelong Learning Programme



THIS COURSE BOOK EXAMINES THE CONCEPTS RELATED TO FREE SOFTWARE BUSINESS AND ALLOWS US TO KNOW AND IDENTIFY CONCEPTS RELATED TO ITS PRODUCTION.

THE STUDENT SHOULD BECOME FAMILIAR WITH AGENCIES AND PROJECTS RELATED TO THE IMPLEMENTATION OF FREE SOFTWARE IN THE PUBLIC SECTOR, PRIVATE SECTOR AND KNOW, IDENTIFY AND UNDERSTAND THE CONSEQUENCES OF THE USE AND EXPLOITATION OF FREE SOFTWARE IN DIFFERENT AREAS OF IMPLEMENTATION.

THIS BOOK ALSO STUDIES HOW TO APPROACH AND IMPLEMENT PROJECTS BASED ON FREE SOFTWARE, PROJECTS TO MIGRATE, PRODUCTION AND DEVELOPMENT METHODOLOGIES AND BUSINESS PROJECTS RELATED WITH FREE SOFTWARE

freeknowledgeinstitute
UOC Universitat Oberta de Catalunya www.uoc.edu
www.ou.nl
UNIVERSITETET I AGDER